USCIPI Report 1050

12

UNIVERSITY OF SOUTHERN CALIFORNIA
IMAGE UNDERSTANDING RESEARCH

Final Technical Report

Ramakant Nevatia
Alexander A. Sawchuk
Principal Investigators
(213) 743-5506

Covering Research Activity During the Period
1 April 1981 through 30 September 1981

30 September 1981

Image Processing Institute
University of Southern California
University Park
Los Angeles, California 90007

Sponsored by

Contract No. F-33615-80-C-1080
DARPA Order No. 3119

IPI

IMAGE PROCESSING INSTITUTE

ENGINEERING

USCIPI Report 1050

(12)

IMAGE UNDERSTANDING RESEARCH

Final Technical Report

Covering Research Activity During the Period
1 April 1981 through 30 September 1981

R. Nevatia
A.A. Sawchuk
Principal Investigators
(213)743-5506

Image Processing Institute
University of Southern California
University Park
Los Angeles, California 90007
30 September 1981

This research was supported by the Defense Advanced Research Projects Agency and was monitored by the Air Force Wright Aeronautical Laboratories under Contract F-33615-80-C-1980, ARPA Order No. 3119.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER USCIPI Report 1050	2. GOVT ACCESSION NO. AD-A110 746	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) IMAGE UNDERSTANDING RESEARCH		5. TYPE OF REPORT & PERIOD COVERED Final Technical Report 1 April 81-30 Sept. 81 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (Principal Investigators) Ramakant Nevatia Alexander A. Sawchuk		8. CONTRACT OR GRANT NUMBER(s) F-33615-80-C-1080
9. PERFORMING ORGANIZATION NAME AND ADDRESS Image Processing Institute University of Southern California University Park, Los Angeles, Ca. 90007		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS DARPA Order No. 3119
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, Virginia 22209		12. REPORT DATE 30 September 1981 13. NUMBER OF PAGES 137 pages
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Air Force Wright Aeronautical Laboratories Wright-Patterson Air Force Base Dayton, Ohio 45433		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) APPROVED FOR RELEASE; DISTRIBUTION UNLIMITED		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) image analysis, feature extraction, statistical and structural texture analysis, symbolic image representation, image to map correspondence, image understanding, image matching, building detection, texture gradient, VLSI architectures		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) This technical report summarizes the image understanding and VLSI system research activities performed by the University of Southern California and the Hughes Research Laboratories during the 1 April 1981 through 30 September 1981 under contract number F-33615-80-C-1080 with the Defense Advanced Research Projects Agency, Information Processing Techniques Office. This contract is monitored by the Air Force Wright Aeronautical Laboratories,		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Wright-Patterson Air Force Base, Dayton, Ohio.

The purpose of this research program is to develop techniques and systems for understanding images, particularly for mapping applications. The research activity includes low level image analysis and feature extraction, statistical and structural texture analysis, symbolic image representations, and image to map correspondence using relational structures. Additional activity is concerned with VLSI architectures for implementation of image analysis and image understanding operations.

"The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government."

ABSTRACT

This technical report summarizes the image understanding and VLSI system research activities performed by the University of Southern California and the Hughes Research Laboratories during the period 1 April 1981 through 30 September 1981 under contract number F33615-80-C-1080 with the Defense Advanced Research Projects Agency, Information Processing Techniques Office. This contract is monitored by the Air Force Wright Aeronautical Laboratories, Wright-Patterson Air Force Base, Dayton, Ohio.

The purpose of this research program is to develop techniques and systems for understanding images, particularly for mapping applications. The research activity includes low level image analysis and feature extraction, statistical and structural texture analysis, symbolic image representations, and image to map correspondence using relational structures. Additional activity is concerned with VLSI architectures for implementation of image analysis and image understanding operations.

TABLE OF CONTENTS

1. IMAGE UNDERSTANDING PROJECTS

1.1. RESEARCH OVERVIEW

This is the final semi-annual report of our research under contract F-33615-80-C-1080. Our research includes work at many levels of an Image Understanding system, including low level feature extraction, symbolic descriptions and image matching. We have also been working with Hughes Research Laboratories on hardware implementation of IU algorithms using VLSI technology. The research results are summarized below.

Image Understanding Projects

Much of our work has concentrated on the matching of a map to an image or one image to another image. An application to the problem of control point location was performed as part of a study conducted by the Defense Mapping Agency. Our results are summarized in Section 1.2.

Another technique of matching map descriptions to image descriptions that primarily relies on line matching is described in Section 1.3. Good results are obtained for tasks such as location of roads, rivers, and buildings in DMA supplied images of the Ft. Belvoir area. We have also applied same techniques

for object detection to Synthetic Aperture Radar (SAR) images. These results are given in Section 1.4. Another approach using probabilistic relaxation is given in Section 1.5 and is a summary of Bir Bhanu's thesis, which is published as a separate report.

Initial attempts at locating objects, such as buildings, in aerial images without a map are described in Section 1.6. Such processing is necessary for change detection.

Applications of our previously described structural texture description techniques to the problems of texture recognition and inference of surface orientation from texture gradients are presented in Section 1.7. This section contains a summary of parts of Felicia Vilnrotter's Ph.D. thesis, also published as a separate report.

Section 1.8 contains a description of texture synthesis techniques using piecewise-linear models, an extension of previous linear techniques for texture synthesis. A comparison of results with previously developed methods is given. The Ph.D. thesis of D.D. Garber, published as USCIPI Report 1000, contains details and comparisons of all the texture synthesis methods developed.

Hardware Development

Our hardware development work with the Hughes Research Laboratories has concentrated primarily on the development of a

residue arithmetic based programmable processor called RADIUS. Other activities have included development of tools for design automation and preliminary definitions of a local area logic processor.

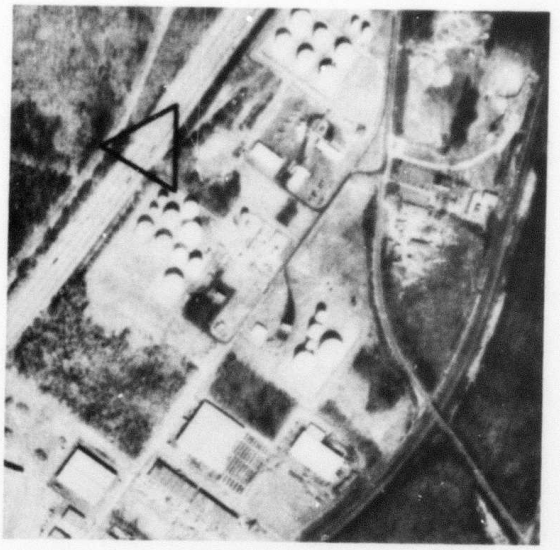
The RADIUS processor can be programmed to perform a variety of local arithmetic operations. Our initial task will be to use it for computing 5X5 convolutions common to many low level processing tasks. A custom circuit has been fabricated and is being tested. Also, this processor has been designed to interface with a PDP-11 UNIBUS.

1.2 SYMBOLIC MATCHING APPLIED TO DMA CONTROL POINT LOCATION

K.E. Price

This report presents the results of applying our relaxation based scene matching system [1] to a new domain - automatic matching of pairs of images. The task was part of a study by the Defense Mapping Agency (DMA) to evaluate various image matching techniques. Because our matching system was designed to work with entire objects (in aerial images - roads, buildings, etc.) we could not perform the task as given finding small sub-images containing parts of objects (corners of buildings) within the large image. But we did demonstrate the ability of our system to automatically segment, describe, and match two images producing a set of objects which can be used as control points.

The two images, shown in Fig. 1, as a substance from a pair of consecutive mapping photos (the black triangle is an artifact on the image). Figure 2 shows the region based segmentation results using only intensity information where very dark and very bright regions may be as small as 100 points and others must be at least 1000 points [2]. These segments form the symbolic representation of the 2 images which is used for the matching



(a) Image 1

(b) Image 2

Figure 1.

(a) Image 1

(b) Image 2

Fig. 2. Segmentation-outlines of regions from the region-based segmentation. The region is the area inside the boundary, except for any holes.

[1].

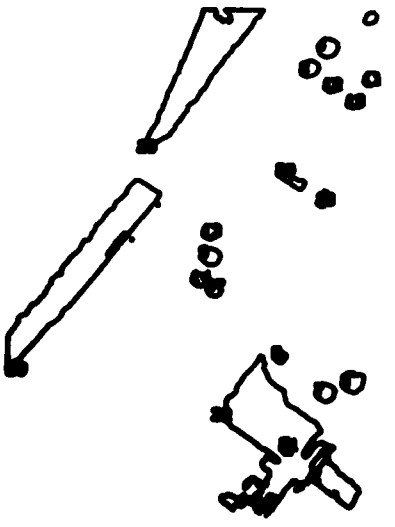
The matching is performed in 2 steps, the first was image 1 as the model and finds the regions in image 2 which correspond to the regions in image 1. The second step in the reverse (switch the roles of image 1 and image 2). Only the regions which match the same in both steps are retained as valid matches. Figure 3 shows the results for matching when image 1 is the model (the labels in the 2 views correspond to the region numbers assigned in the segmentation of image 1). Figure 4 is the results for the second step. Figure 5 shows the final valid control point regions (with labels from image 1). Table 1 give the disparities for each of the regions (in pixels). The region locations are computed as the center of the mass of the pixels in the region.

REFERENCES

[1] O.D. Faugeras and K.E. Price, "Semantic Description of Aerial Images Using Stochastic Labeling," IEEE Trans. on Pattern Recognition and Machine Intelligence, Nov. 1981.

[2] R. Ohlander, K. Price and R. Reddy, "Picture Segmentation Using a Recursive Region Splitting Method," Computer Graphics and Image Processing, Vol. 8, 1978, pp.313-333.

(a) Image 1



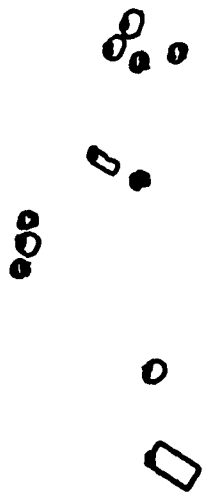
(b) Image 2

Fig. 3. Matching: Given Image 1 as a model find the objects in Image 2.

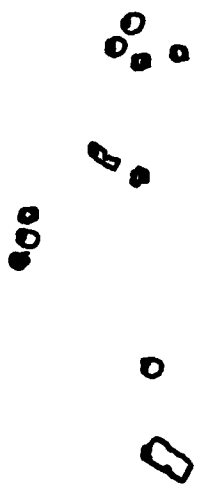
(a) Image 2

(b) Image 1

Fig. 4. Matching: Given Image 2 as a model find the objects in Image 1.



(a) Image 1



(b) Image 2

Fig. 5. Matching: The matches which were the same in the results in Figs. 3 and 4.

Table 1. Disparities for the Matched Regions

Object label (Image 1 number, Image 2 number)	Location in Image 1 (Row, Column) or (Scanling position on line)		Location in Image 2 (Row, Column)		ΔRow, ΔColumn	
2,2	296.1	87.3	293.8	79.1	2.3	8.2
4,5	273.4	494.3	278.7	479.6	-5.3	14.7
6,6	244.5	368.8	247.6	358.5	-3.1	10.3
7,7	282.6	174.3	283.0	163.0	-0.4	11.3
8,9	257.3	473.2	262.5	459.4	-5.2	13.8
9,8	174.1	295.0	177.7	285.5	-3.6	9.5
11,13	314.2	466.1	319.1	451.0	-4.9	15.1
12,12	278.5	459.0	283.4	444.8	-4.9	14.2
13,15	173.2	317.4	176.9	307.4	-3.7	10.0
15,19	164.4	273.1	167.3	264.1	-2.9	9.0
16,16	275.2	351.0	278.3	340.0	-3.1	9.0

1.3 MATCHING OF A MAP WITH AN AERIAL IMAGE

Gerard G. Medioni

Abstract

Discrete relaxation is applied to the problem of matching linear features extracted from an aerial image with a set of linear features derived from a map. The method is adapted to take into account the inaccuracy inherent to real world data. It is first used to find the most prominent features in the low resolution version of the image (rivers, large roads), then to find man made local structures such as buildings in full resolution windows of the image. This relaxation algorithm departs substantially from previous methods because of the non-symmetric character of the relations. Very promising results are shown and extensions are outlined.

INTRODUCTION

Suppose we are given a very high resolution aerial picture with a known orientation and a known altitude, together with a detailed map of the area. How can we determine which parts of the picture correspond to given elements of the map?

This is a very complex problem for many reasons, the main one being that the picture is described in term of pixel intensities when the map is a set of high level abstracted objects represented in a schematic form and with implicit geometric relations with each other. Therefore, we need to change the representation of both the picture and the map knowledge to make them identical or at least comparable.

There are two extreme "solutions" that we shall reject:

1) Represent the map in terms of an intensity array and compare it to the picture. The problem then becomes equivalent to change detection between two pictures, which has been treated with variable success by many researchers, but here the transform map image would require knowledge of illumination conditions and reflectance maps and models of the region, which are not readily available.

2) Represent the image in the same terms as the map (road, buildings, rivers, lakes,...) and perform some kind of subgraph isomorphism, alas the day is not near when we are able to successfully segment an image into constituting semantic components.

Instead, we first extract the edges from the picture, along with the strength and orientation information. We follow this with a linear approximation of these edge elements. The edge detector used is either six 5x5 masks with different orientation

convolved with the image, or a Laplacian-Gaussian mask suggested by Marr [1] and implemented in the USC linear feature extraction system [2,3].

The two different features we consider are called SEGMENTS and APARS. Segments are linear pieces approximating a set of edge points. Apars are a representation of 2 segments running in two opposite directions and separated by a known width. They are very convenient for representing roads and rivers. The process of encoding pieces of the map in terms of these linear features is then an easy task performed manually so far. With now both the map and the image in the same format, we can formulate the problem as: what elements, in the model do the elements in the picture correspond to, if any, based on geometrical constraints. A previous paper [4] dealt with local preprocessing of the input in order to remove small edges and to fill gaps in long features and will not be described again here.

The input we will consider is then obtained as shown in Fig. A. The next section provides definitions of the terms and description of the method. The third section is concerned with implementation, the fourth discusses results and possible extensions.

DEFINITIONS AND DESCRIPTION OF THE METHOD

Assumptions

- The model and image have approximately the same orientation.

  This is justified by the fact that the map is oriented and that the orientation of the plane at the time of the picture is registered with it.

- The scale factor from the model to the image is known, we shall call it μ.

  This is justified because the camera characteristics are known and the altitude of the plane is known at the time of the picture.

Definitions

We will denote the linear features of the image as a_i, i in [1..n] and call them objects. We will denote the linear features of the object as λ_j, j in [1..m] and call them labels. We are interested in computing the quantity p(i,j) which is the possibility for object a_i to have label λ_j. $\forall$ (i,j) in [1..n, 1..m] p(i,j) in [0,1]. This quantity is NOT a probability for the following reasons:

- It is possible for an object a_i to have no possible label λ_j and we did not define the "no label" as a label, and hence p(i,j) can be 0.

- It is possible for several objects to have the same label λ_j due to some fragmentation occurring during segmentation,

Therefore $\sum_i p(i,j)$ is not restricted.

- Finally, an element a_i may have several labels, therefore $\sum_j p(i,j)$ may be greater than one.

Method

Rough Description:

The idea is the following: first, assign possibilities based on angular orientation, then at each step, assign a label λ_j to an element a_i and see if the rest of the picture "fits" the model to accept or reject this label. Iterate until you reach a stable configuration.

Formal Description

(α) Initial Assignments of Possibilities

Let θ_i be the angular orientation of a_i, let ϕ_j be the angular orientation of λ_j.

Then

$$\forall(i,j) \text{ in } [1..n,1..m],\ p^0(i,j) = 1 \text{ if } |\theta_i - \phi_j| \leq 15^\circ$$
$$= 0 \text{ otherwise}$$

(β) Window Definition

Any object a_i can be represented by a vector $\overrightarrow{A_iB_i}$. Any label λ_j can be represented by a vector $\overrightarrow{P_jQ_j}$. Assigning label λ_j to object a_i means that we should find objects

a_k in the position where one would expect to find the other labels of the model. So, for each label λ_k different from λ_j, we define a window w(i,j,k) in which we look for an object a_k with possible label λ_k to confirm the labeling of object a_i as λ_j.

(γ) Window Description

We consider object a_i $(\overrightarrow{A_iB_i})$ with possible label $\lambda_j(\overrightarrow{P_jQ_j})$ and we look for a_h with label $\lambda_k(P_kQ_k)$.

To define w(i,j,k) we consider 2 extreme cases:

- First we identify A_i with P_j to define the points

$$\overrightarrow{OR_1} = \overrightarrow{OA_i} + \mu\overrightarrow{P_iP_k}$$
$$\overrightarrow{OS_1} = \overrightarrow{OR_1} + \mu\overrightarrow{P_kQ_k}$$

- Then we identify B_i with Q_j to define the points

$$\overrightarrow{OR_2} = \overrightarrow{OB_i} + \mu\overrightarrow{Q_iP_k}$$
$$\overrightarrow{OS_2} = \overrightarrow{OR_2} + \mu\overrightarrow{P_kQ_k}$$

This defines the rectangular window w(i,j,k) with the points R_1, R_2, S_1, S_2 as the four corners. An example is shown in Fig. B.

With each window w(i,j,k), we associate a characteristic function updated at each iteration, $\delta^t(i,j,k)$ defined by

$\delta^t(i,j,k)$ = 1 if there is an element a_h with label λ_k in w(i,j,k) at iteration t

= 0 otherwise.

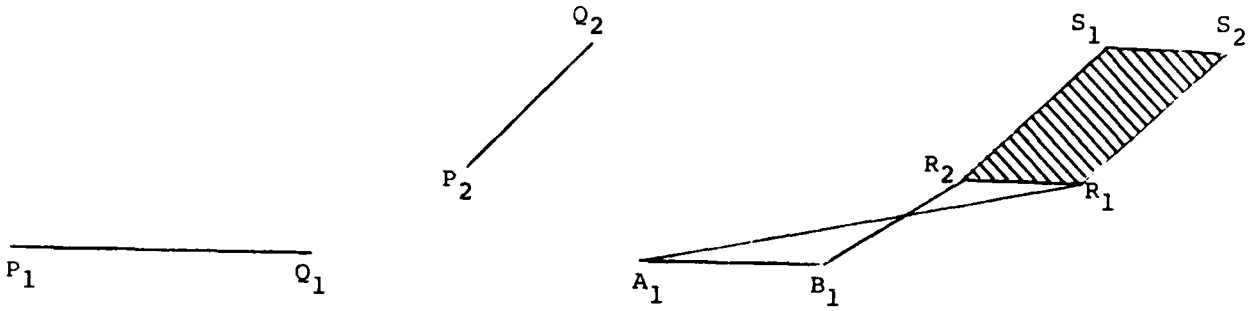

Q_2
P_2
P_1
Q_1
Model
S_1
S_2
R_2
R_1
A_1
B_1
Image

Fig. A

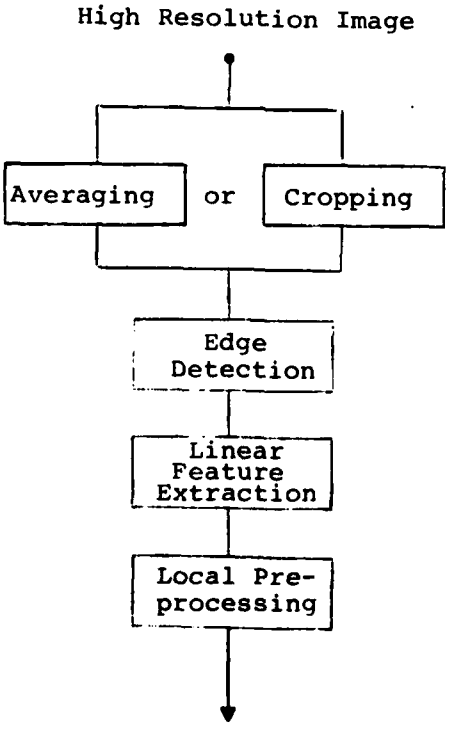

High Resolution Image
Averaging
or
Cropping
Edge Detection
Linear Feature Extraction
Local Pre-processing
Input to Matcher

Fig. B.

(δ) Core of the Method

- First Pass

At each iteration, the quantity $p^t(i,j)$ is derived from the following formula

$$\forall(i,j) \text{ in } [1..m,1..n],\ p^{(t+1)}(i,j) = 1 \text{ if } \left[1+ \sum_{\substack{s=1 \\ s\neq j}}^{m} \delta^t(i,j,s)\right] \geq q \text{ and } p^t(i,j) = 1$$

$$= 0 \quad \text{otherwise.}$$

The quantity q is a variable provided by the user and is a measure of the way model and image agree. Setting q to m(the number of labels in the model) suggest that we expect a perfect match that is a one to one or many to one mapping from the image into the model.

Setting q to a low percentage of m is rather meaningless since a lot of elements will be labeled, even though they are very partially matched. The stopping criterion is simply that

$$\forall(i,j) \quad p^{(t+1)}(i,j) = p^t(i,j)$$

Claim 1

This process converges in a finite number of iterations.

Proof

The way the method is presented, at each iteration after the initialization, an element cannot acquire a new label but only

discard an existing one. Let N^t be the total number of labels attached to all the elements.

$$\forall t,\ N^{(t+1)} \leq N^t \quad \text{and} \quad \forall t \quad N^t \geq 0$$

The sequence of positive integers N^t is decreasing and has 0 as a lower bound, hence it converges to a limit N_ℓ. If $N^{(t+1)} = N^t$ then $N^t = N_\ell$ since it means that no labels have been removed during the (t+1) iteration. If N_ℓ is 0, then the pattern in the model has no match in the image, otherwise we found a stable configuration. The algorithm converges in less than $N^{(0)}$ iterations since $0 \leq N_\ell \leq N^{(0)}$ and the numbers in the sequence $N^{(0)}, N^1, \ldots, N_\ell$ are all different.

- Second Pass

So far, the method used here is a slightly modified version of the basic discrete relaxation technique described by Rosenfeld [5]. It provides multi-arc consistency without being as powerful as the scheme proposed by Freuder [6]. However, the results after this first pass sometimes accept labels for an object out of place because this object is supported by the correctly labeled objects. To take care of these effects, we use the fact that our relations are not symmetric: a_k in $w(i,j,k)$ does not imply a_i in $w(h,k,j)$. So, we use what we call "reverse compatibility", that is, given that a_i has label λ_j, how many object does a_i support? If it does not support at least q objects, we can remove it at no

cost.

Formally, at each iteration t, we compute the function $\alpha(i,j,k,\ell)$:

$\alpha^t(i,j,k,\ell)=1$ if a_k in $w(i,j,\ell)$ and $p^t(i,j) = 1$

$=0$ otherwise.

and now

$$p^{(t+1)}(k,\ell)= 1 \text{ if } \left[1+\sum_{\substack{i=1\\ i\neq k}}^{n}\sum_{\substack{j=1\\ j\neq \ell}}^{m}\alpha^t(i,j,k,\ell)\right] \geq q$$

$$= 0 \text{ otherwise.}$$

Once again we stop when $\forall(i,j)$, $p^{(t+1)}(i,j) = p^t(i,j)$

Claim 2

This process converges in a finite number of iterations.

Proof

Identical to proof of claim 1.

IMPLEMENTATION

A computer program in the programming language SAIL was written to implement the previously described method. However, variations were made to improve the running time:

- We do not compute and store the values of $\delta(i,j,k)$ because it is a sparse array and because the updating of $\delta(i,j,k)$ at each

iteration requires a lot of bookkeeping. Instead, we scan the window $w(i,j,k)$ until we find an object with the desired label, if any.

- If at iteration t we find that $p^{(t+1)}(i,j) = 0$, we set $p^{(t)}(i,j)$ to 0 immediately.

Claim 3

This modification does not alter the final solution.

Proof

- If $\delta^{(t+1)}(i,j,s) = \delta^{t}(i,j,s)$ then obviously replacing one by the other does not change anything.
- If $\delta^{(t+1)}(i,j,s) = 0$ and $\delta^{t}(i,j,s) = 1$ there are 2 cases:

  1) $p^{(t+1)}(i,j)$ is the some computed with δ^{t} or $\delta^{(t+1)}$. Then again it does not change anything.

  2) $p^{(t+1)}(i,j)$ becomes 0 when computed with $\delta^{(t+1)}$. Then at the next iteration, $p^{(t+2)}(i,j)$ would have become 0, so setting it to 0 at iteration t+1 simply speeds up the convergence rate.

- Degenerate Cases.

  There are two of them which will be treated in the same way:

  a) the length of $\overrightarrow{A_i B_i}$ and $\overrightarrow{P_j Q_j}$ are the same.

b) $\lambda_j(\overrightarrow{P\ Q}\ )$ and $\lambda_k\ (\overrightarrow{P_k Q_k})$ have exactly the same orientation.

In these two cases, the rectangular window becomes a single line $\overrightarrow{UV}$, allowing no errors between image and model, not even a single pixel. When this happens we simply generate a rectangular window, one of the axis being $\overrightarrow{UV}$, the other one a vector normal to $\overrightarrow{UV}$ of length 4. This takes care of small errors at the time of generating the model.

RESULTS

We worked on the image DMA3 representing part of the Fort Belvoir Military Reservation in Virginia. The original picture, shown on Fig. 1 has a resolution of 2048 x 2048 pixels.

Figure 2 represents a picture of the part of the map corresponding to the DMA3 image. As we can see on Fig. 1, the original image is very detailed, and in order to segment it we proceed hierarchically: to find the most prominent features such as large roads and rivers, we use a smoothed version of the original image. This is shown in Fig. 3 which has a resolution of 256x256 pixels.

Now, as explained in the introduction, we extract the linear features from the picture in Fig. 3.

Fig. 1. High resolution (2048x2048) image.

Fig. 2. Map of the region corresponding to the image in Fig. 1.

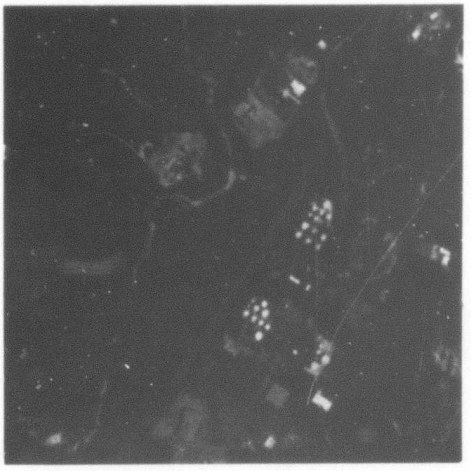


Fig. 3. Low resolution (256x256) version of the image in Fig. 1.

Fig. 4. Segments extracted from the image in Fig. 3.

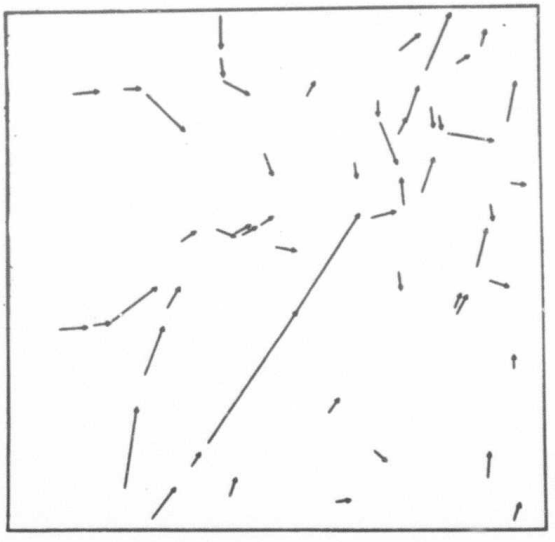


Fig. 5. Apars extracted from the image in Fig. 3, and filtered.

Figure 4 shows the oriented segments; as we can see, most of the small details have disappeared. Since we are interested in roads and rivers, we extract the apars having a maximum width of 8 pixels and filter out the very small apars.

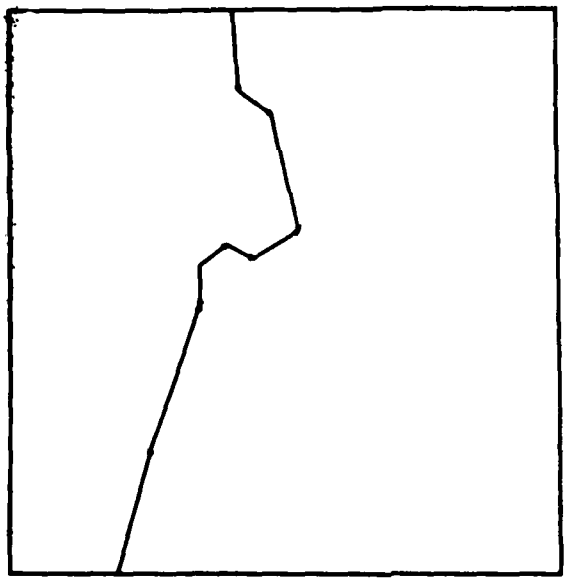
The result is shown in Fig. 5, and we are applying our matching algorithm on the corresponding file.

By looking at the map, we know that the major features we expect to find are the largest road, denoted as highway 95 on the map, and the river called Accotink river on the map, so we hand generate a piecewise linear approximation of these two features, that is a set of apars representing them. Figure 6a represents the model of the river and Fig. 7a represents the model of the road.

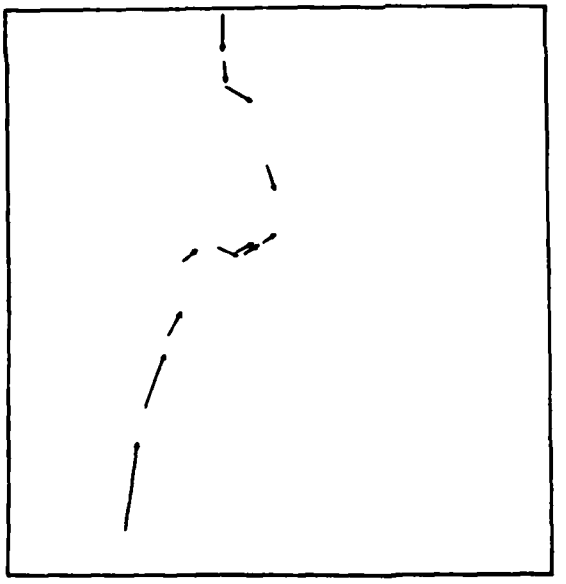
We now use our matching algorithm to find which elements in Fig. 5 correspond to which part of the hand generated model description.

Figure 6b represents the apars that have been successfully identified as corresponding to the description of the river and Fig. 7b represents the one identified as corresponding to the description of the road. The results obtained here are similar to the ones expected from a human observer.

Once the prominent features are identified in the low resolution image, it becomes easy to compute better estimates of the scale and orientation changes between image and map, one

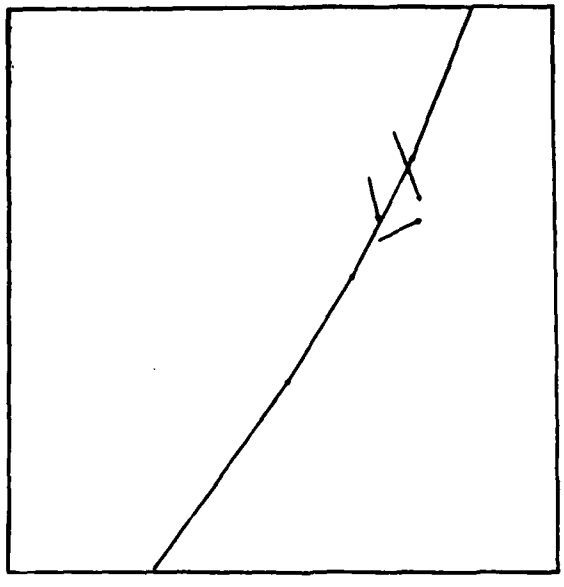


a) Hand generated model of the river.

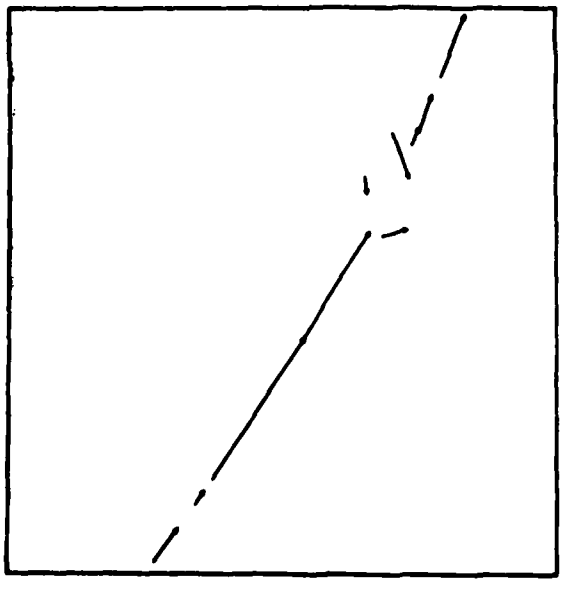


b) Apars from Fig. 5 identified with the river as described in the model in Fig. 6a.

Fig. 6. Matching apars.



a) Hand generated model of the road.



b) Apars from Fig. 5 identified with the road as described in the model in Fig. 7a.

Fig. 7. Matching apars.

method being described in [6,7]. This step has not been implemented so far, but only hand simulated.

We now can concentrate on the details of the image by going back to the full resolution image. Since we established a good correspondence between image and map, we now are able to define for each object in the map, such as buildings, a small window in which we are certain to find this specific object, if present.

We present two examples of matching using this scheme:

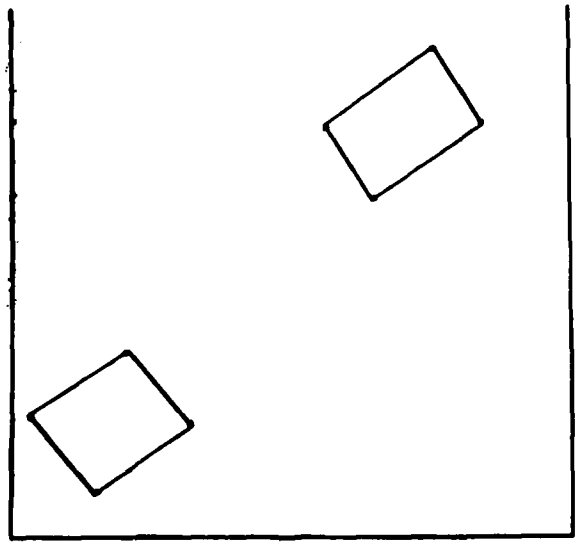
1) Figure 8a represented the window where we located two known buildings. These appear clearly to a human observer because the picture is quite crisp, no sides are missing and there are not too many segments.

   Figure 8b is the hand made model of these two buildings. Their shape has been voluntarily distorted slightly to show that the method is not sensitive to such small variations.

   Figure 8c represents all the segments from Fig. 8a that have been assigned a label after the first pass, and we can see that two of these segments have been assigned a label even though they do not belong to the buildings.

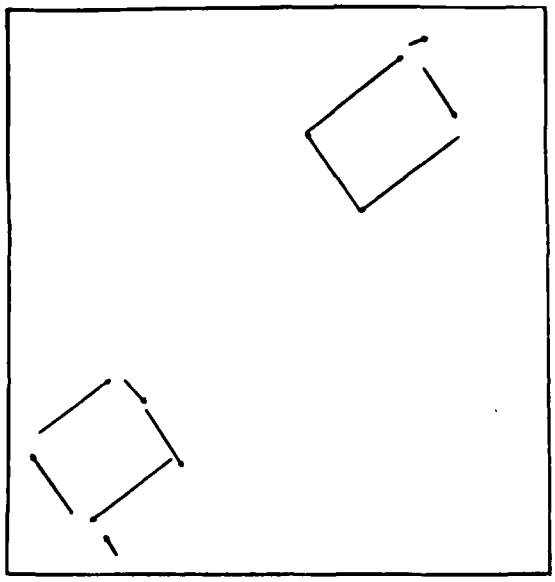
   Figure 8d represents all the segments that are matching the model at the end of pass 2. The second pass efficiently removed the two incorrectly matched segments because they do not support the others, that is they do not confirm the

a) Segments extracted from a window of the high resolution image in which we expect to find 2 given buildings.

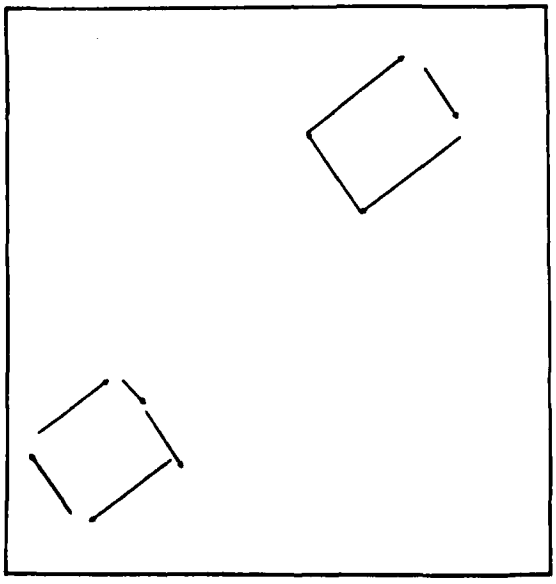


b) Model representing the 2 given buildings.

Fig. 8. Matching segments.



c) Segments from Fig. 8a that have been matched with the model at the end of pass 1.



d) Segments from Fig. 8a that have been matched with the model at the end of pass 2.

Fig. 8. Continued.

labeling of any other segments.

2) Figure 9a represents a window in which we located the two top most buildings. It is a very dense picture in terms of number of segments, furthermore it is of great interest because the leftmost wall of the leftmost building is not found by the edge detector, and therefore the match with the model will be imperfect.

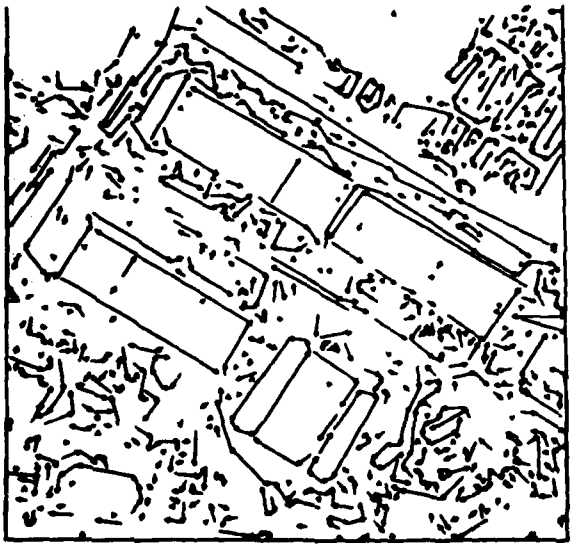
Figure 9b represents the hand generated model of the two buildings. Once again, the model has been drawn with only moderate accuracy to show the robustness of the method.

Figure 9c represents all the segments from Fig. 9a that are matched with a segment in the model at the end of the first pass. We can observe that no segment has been matched with the missing edge of the building and that once again two segments have been matched that do not belong to any building.

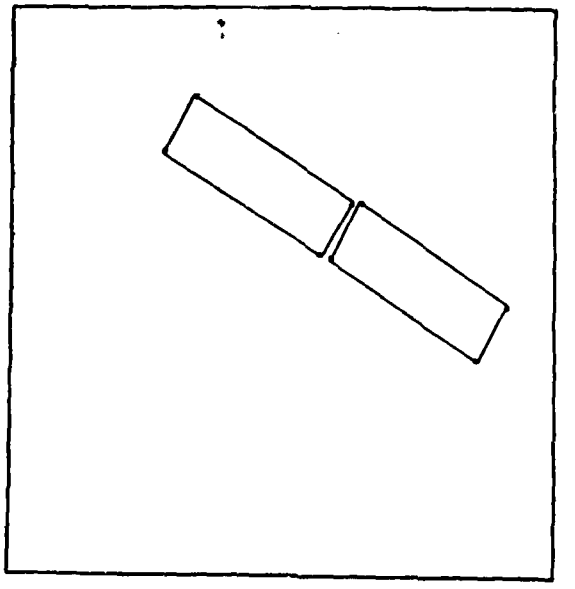
Figure 9d shows that these two segments have been successfully removed by the processing in the second pass because they are "parasites": They are supported by the correct segments but do not support them in return.

CONCLUSION

This paper describes a successful application of the

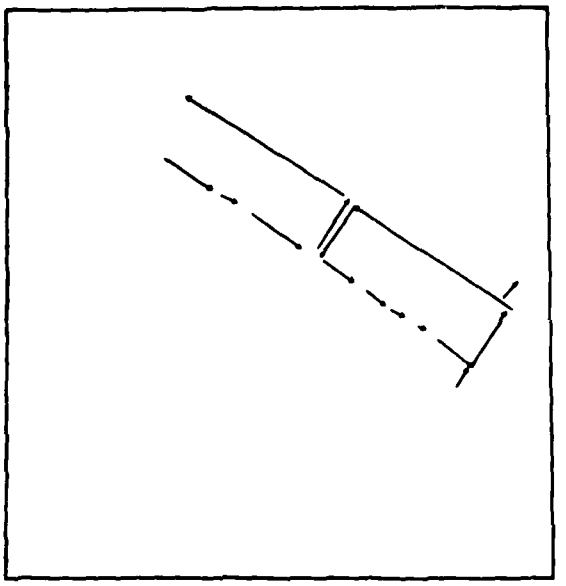


a) Segments extracted from a window of the high resolution image in which we expect to find 2 given buildings.

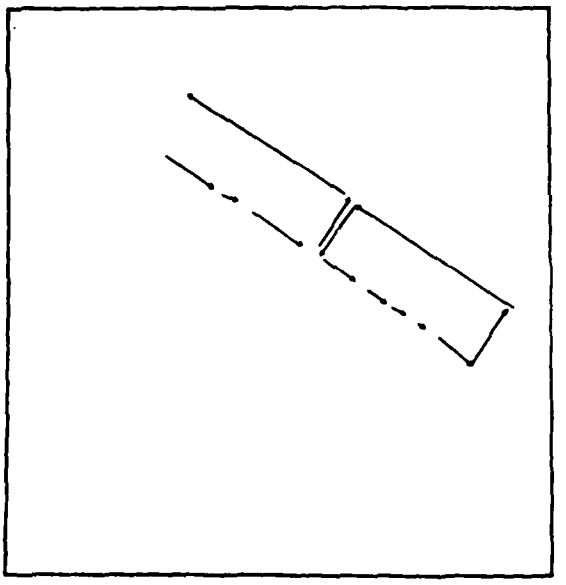


b) Model representing the 2 given buildings.

Fig. 9. Matching segments.



c) Segments from Fig. 9a that have been matched with the model at the end of pass 1.



d) Segments from Fig. 8a that have been matched with the model at the end of pass 2.

Fig. 9. Continued.

relaxation process to matching an image with a known description of it. Related work can be found in [8,9]. Limitations of the method are that if the model is too poor, that is, contains very few elements, it is very likely that the program will find a lot of possible matches with each label in the model. This phenomenon is also due to the fact that we do not take the length of the linear features into account (to respect the fragmentation segmentation that occurs when we extract them), hence voluntarily reducing the knowledge we have of the scene. Along the same line, if the image is very dense in terms of linear features and has not been preprocessed, some unwanted match may appear due to the very short linear features.

We can extend this method to problems where the orientation is not know either by rotating the model based on histogram comparison, or by trying to match with a set of models, each one being rotated from the other by a 15° angle, or even by assigning all possible labels to all possible objects to start with. We can also extend it to the detection of an occluded object in a simple environment, such as a bin of auto parts, and eventually to the tracking of an object through a sequence of frames.

REFERENCES

[1] D. Marr, "Early Processing of Visual Information", Phil. Trans. Roy. Soc. B. 275, 1976.

[2] R. Nevatia and K. Ramesh Babu, "Linear Feature Extraction and Description," Computer Graphics and Image Processing, Vol. 13, June 1980, pp. 257-269.

[3] A. Huertas and R. Nevatia, "Edge Detection in Aerial Images Using G(x,y)," USCIPI 1010, April 1981.

[4] Gerard G. Medioni, "An Algorithm to Group the Edges of a Picture Using a Local Criterion," USCIPI 1010, April 1981.

[5] A. Rosenfeld, R.A. Hummel and S.W. Zucker, "Scene Labeling by Relaxation Operations," IEEE Trans. Syst. Man, Cybern., Vol. SMC-6, pp. 420-433, June 1976.

[6] S.A. Duami, A.L. Luk, J.P. R, C.S. Clark, B.L. Bullock, "Model Based Scene Matching," Research Report 509, Hughes Research Laboratories, Malibu.

[7] C.S. Clark, A.L. Luk, and C.A. McNary, "Feature-Based Scene Analysis and Model Matching," Proceedings of NATO Advanced Study Institute on Pattern Recognition and Signal Processing, Paris, France, June 1978.

[8] E.C. Freuder, "Synthesizing Constraint Expressions," Commun. Ass. Comput. Mach., Vol. 21, No. 11, 1978.

[9] Les Kitchen, "Relaxation Applied to Matching Quantitative Relational Structures," IEEE Trans. Syst. Man, Cybern., Vol. SMC-10, No. 2, February 1980.

1.4 OBJECT DETECTION IN SYNTHETIC APERTURE RADAR IMAGES

J. Burns, A.Huertas and R. Nevatia

INTRODUCTION

This section describes a study to evaluate the effectiveness of some of our image analysis techniques to the tasks of processing Synthetic Aperture Radar (SAR) images. These images are typically characterized by high speckle noise and relatively poor resolution. However, for some applications specific objects are to be located and several cues to use are given *a priori*. The 384x352 SAR images in this study were supplied by the Avionics Laboratory, Wright-Patterson Air Force Base, Ohio.

Our study included the extraction of linear features in the SAR images and matching them to corresponding wire frame models that could be available from a simple map of the area.

The linear feature extraction technique has been previously described in [6]. The local edge detection is performed by convolving the image with a set of step masks or by detecting the zero-crossings in the convolution of the image with a Laplacian Gaussian filter as described in [1] and [2]. The matching technique is described in another section of this report [4].

LINEAR FEATURE EXTRACTION

The linear features in the SAR images studied are generated by objects such as roads, canals and long building structures. High speckle noise and poor resolution result in heavy fragmentation and a high density of texture elements with similar contrast and width to that of the features of interest.

Two directions were investigated. First the use of large convolution masks for local edge detection and second, improving the extracted segments by grouping techniques to provide sufficient feature information to achieve successful matches.

Object boundaries are obtained in the form of line segments and linear features are defined by anti-parallel segments (apars). Apars are pairs of segments of opposing contrast and are labelled bright or dark depending upon the brightness of the feature relative to its background [6].

Successful edge detection is highly dependent on the content of the image. Early experiments showed the need to use convolution masks of larger size than those we have used for conventional aerial images. A minimum size of 9x9 pixels for step masks and 17x17 for Laplacian Gaussian masks proved useful. In some cases the use of rectangular or long strip masks step masks produce marginally superior results but the main problems still remain.

The use of Laplacian Gaussian masks provides for better continuity along the features. On the other hand, small bright regions are detected as oversized blobs if their width is smaller or comparable to the width of the central excitatory region of the filter. This may cause considerable fusion of nearby features with the boundaries of the desired linear features. However, in some cases such fusion is desirable. The clusters of bright small regions corresponding to objects such as buildings and vehicles become fused and their boundaries are more apparent.

A great deal of noisy edges are suppressed during the thinning and thresholding steps if step masks are used. For Laplacian-Gaussian masks, the energy on both sides of a zero-crossing is used as thresholding criteria. If the linear features are defined by anti-parallel (apars) segments, a difficulty arises when only one of the boundaries of a linear feature is distinguishable, as enough apars are not produced. This suggests, in some cases, the processing of the line segments rather than the processing of the apars. The matching technique is able to work with either segments or apars.

The success of our matching technique depends on sufficient number of extended linear features being extracted. Longer linear features can be obtained by segment or apar grouping on collinearity and proximity. Our apar grouping technique follows an algorithm previously reported [3]. For segment grouping we developed a one pass technique that combines segment grouping and

selection, and is described later in this section.

OBJECT DETECTION BY MATCHING

Object detection is accomplished by matching the extracted linear features to corresponding wire frame models that could be available from a simple map of the area. The matching technique is described in another section of this report [4]. We discuss now some of the results obtained.

Figures 1a and 1b show the SAR image CANAL and its corresponding wire frame model. The main features are the two canals which have a high contrast with the background. Both local edge detection techniques produce fairly good and equivalent results. Figures 1c-1e show the line segments obtained using 9x9 step masks, the corresponding grouped dark apars, and the successful result of the match to the wire frame model of Fig. 1b.

Figures 2a and 2b show the BUILDINGS image, which includes groups of building structures, and the corresponding wire frame model. The features are of high contrast and adequate width and length located against fairly low background texture.

Although a set of six rectangular (11x5) step masks leaves the outlines of the buildings fairly distinguishable, their lack of continuity causes few useful apars to be produced. A 23x23 Laplacian Gaussian mask, with a central region 6 pixels wide, was

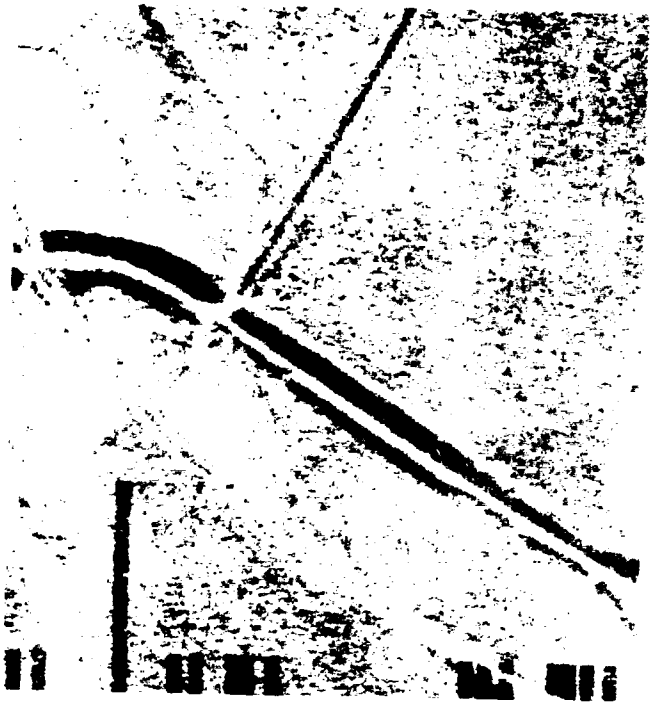


Fig. 1a. Canal.

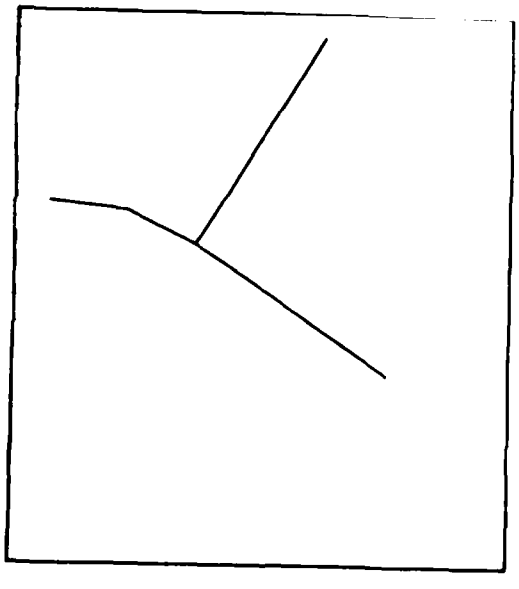


Fig. 1b. Canal Model.

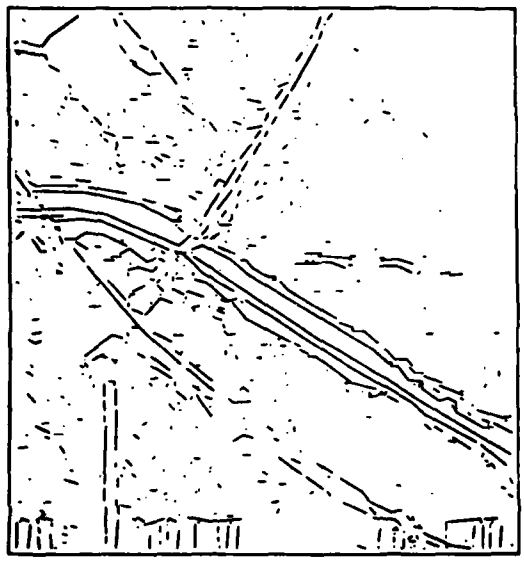


Fig. 1c. Canal segments.

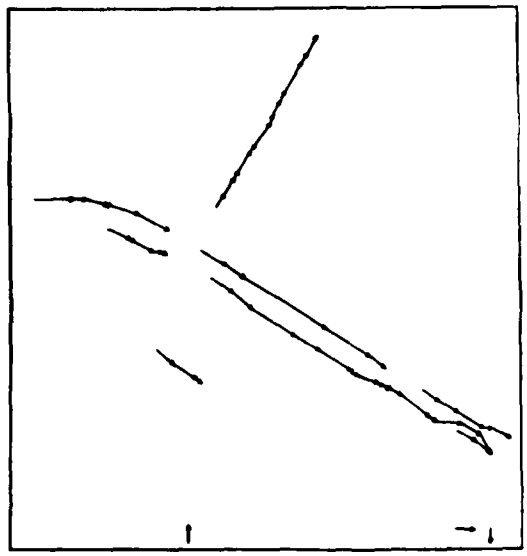


Fig. 1d. Grouped dark apars.

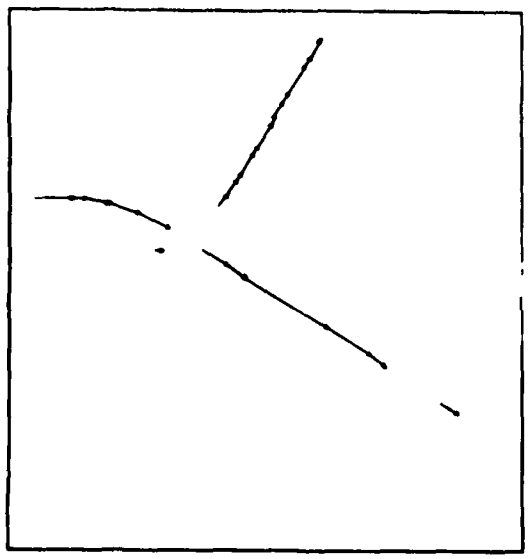


Fig. 1e. Match result.

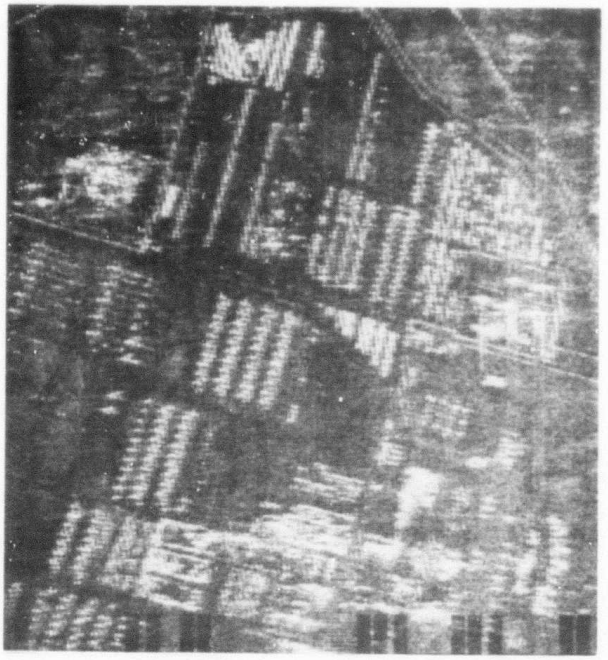


Fig. 2a. BUILDINGS.

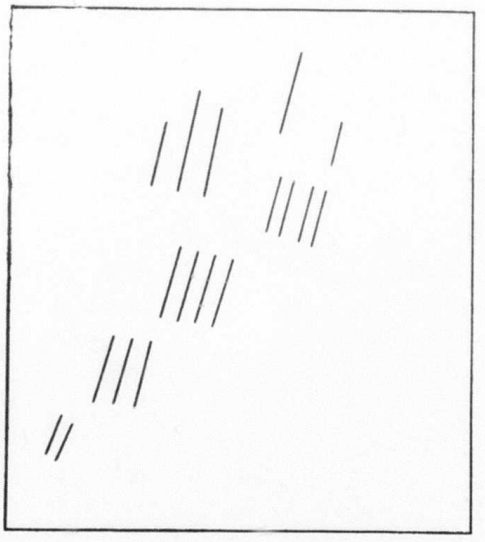


Fig. 2b. BUILDINGS model.

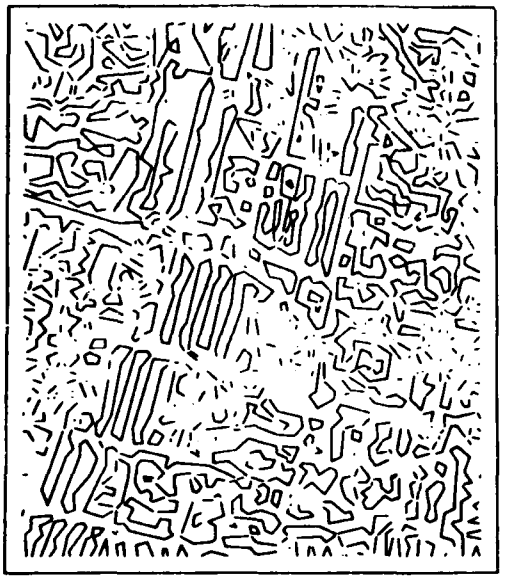


Fig. 2c. BUILDINGS segments.

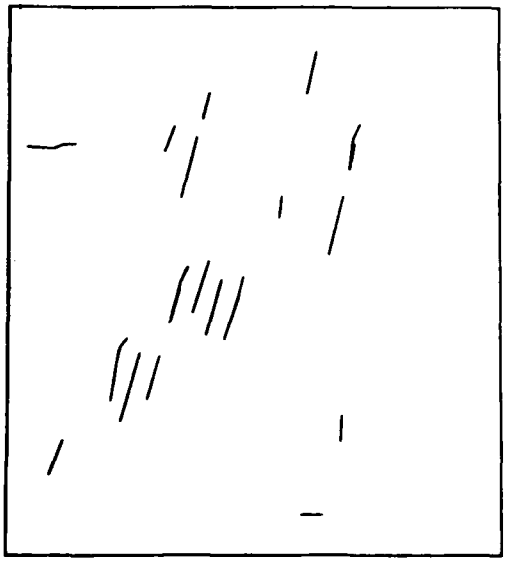


Fig. 2d. Grouped bright apars.

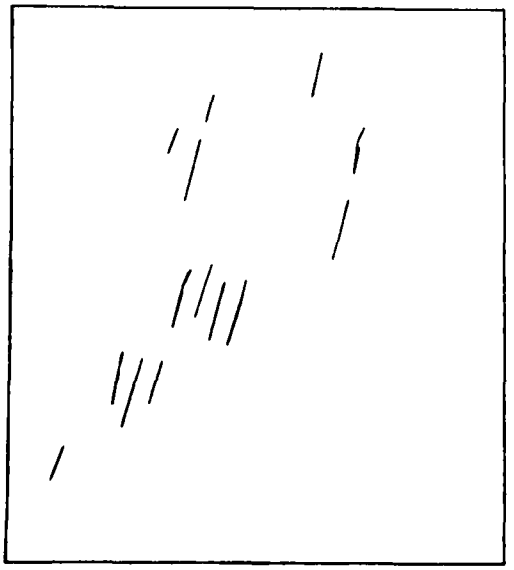


Fig. 2e. Match result.

used to produce the line segments shown in Fig. 2c. Figures 2d and 2e show the grouped bright apars and the successful resulting match to the model of Fig. 2b.

In some cases, low contrast and texture prevent the extraction of good segments and apars. A segment grouping and selection technique was used to improve upon the initial results. Collinearity and gap size are the basic criteria used for segment grouping. Collinearity is defined in terms of lateral and angular disparity between the segments. Referring to Fig. 3a, lateral separation is considered to be the distance (a) between the segment midpoints along the normal to the average of the two segment orientations . The gap size has been defined as the distance (b) between the two nearest endpoints along the axis determined by the segments' orientation. Notice that this measure allows for backward bridging, which can be considered useful in some cases (see Fig. 3b).

The process of pairing the candidate segments before the actual decision ought to reduce the large number of possible tests while still guaranteeing that each genuinely bridgable pair is tested. A scheme for segment classification with respect to orientation [5] was adopted in this case. Classification is followed by sorting the segments within the classes in descending order of lateral position, so that collinear segments tend to be within a short search distance from each other. In order to ensure that all useful pairs are tested, these classes are

overlapped to cover the maximum acceptable angular disparity between segments. Selecting the most desirable bridge is based on the length that the bridged segments would have taken together along the segments' direction as in the pairing of segments (a) and (b) over other possibilities in Fig. 3c. Since segment grouping creates new or longer supersegments (families of segments [6]) their length is used to reduce the size of the set of segments to be matched to the wire frame model.

Figures 4a and 4b show the FIELD image and its corresponding wire frame model. The contrasts of the field boundaries and roads, the principal features of the image, are very low. Figure 4c shows the segments obtained by using a 23x23 Laplacian Gaussian mask with central region of width 6. The above segment grouping technique was applied to these segments with the result shown in Fig. 4d. The result of the match to the model of Fig. 4b is shown in Fig. 4e.

CONCLUSIONS

Our techniques have been successful in locating desired objects in a variety of SAR images in spite of the poor quality and high speckle of these images. The performance of our techniques is dependent on the ability to extract sufficient number of linear features. Our techniques may fail if the scene is very complex and contains no extended linear features.

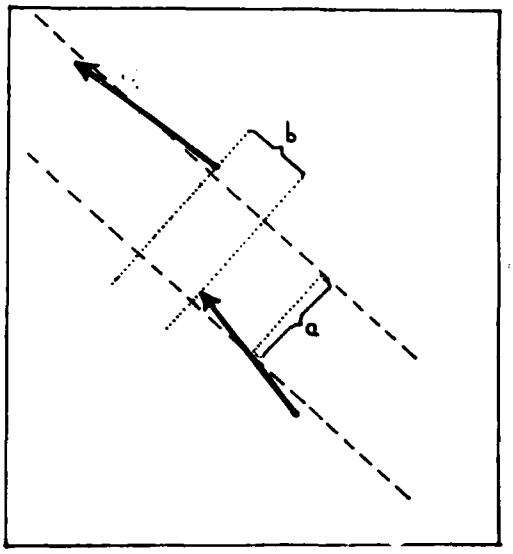


Fig. 3a. Segment Collinearity

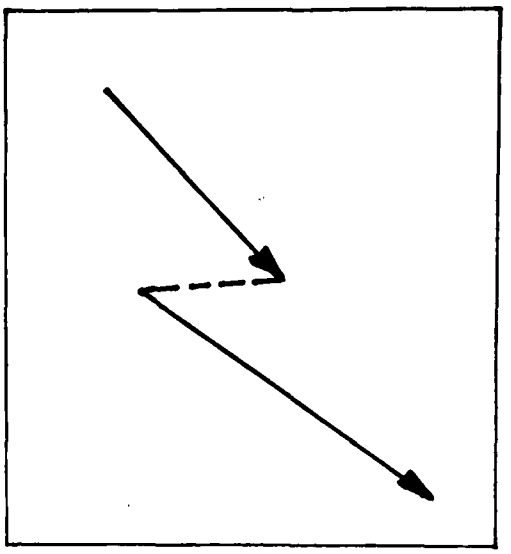


Fig. 3b. Backward bridging.

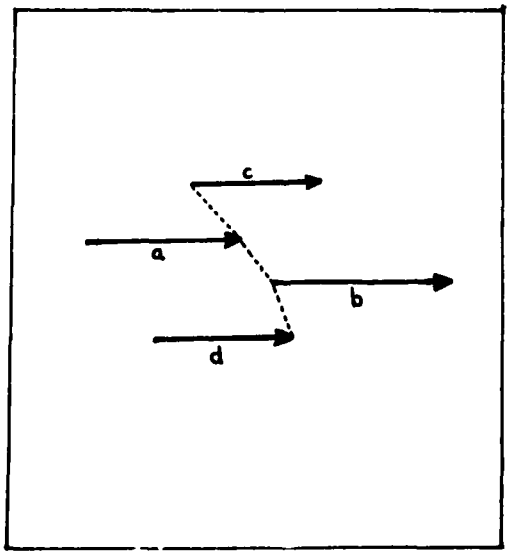


Fig. 3c. Segment pair selection.

Fig. 4a. FIELD.

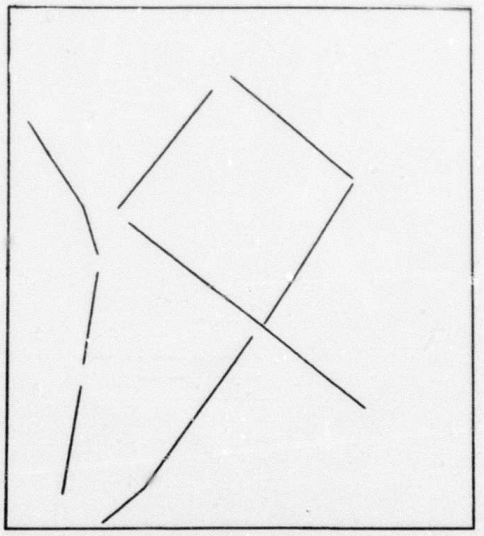


Fig. 4b. FIELD model.

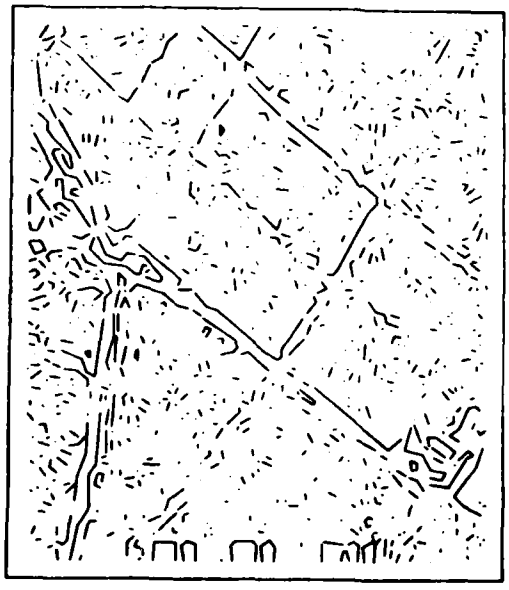


Fig. 4c. FIELD segments.

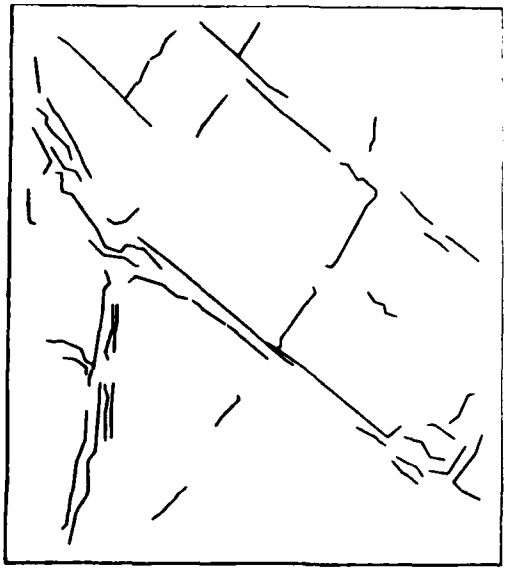


Fig. 4d. Grouped segments.

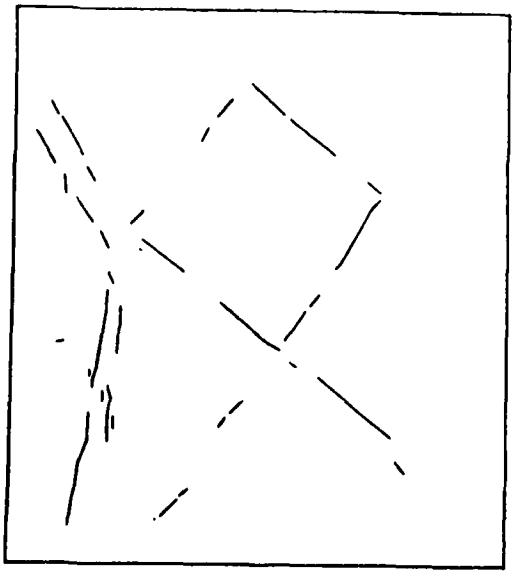


Fig. 4e. Match result.

REFERENCES

[1] A. Huertas and R. Nevatia, "Edge Detection in Aerial Images Using Laplacian-Gaussian Filters," USCIPI Report 1010, April 1981, pp. 16-26.

[2] D. Marr and E. Hildreth, "Theory of Edge Detection," Proc. Royal Soc. London, B207, 1980, pp. 187-217.

[3] Gerard G. Medioni, "An Algorithm to Group Edges Using a Local Criterion," USCIPI Report 1010, March 1981.

[4] Gerard G. Medioni, "Matching of a Map in an Aerial Image," USCIPI Report 1050, September 1981.

[5] R. Nevatia, "Locating Object Boundaries in Textured Environments," IEEE Trans. on Computers, Vol. C-25, No. 111, November 1976, pp. 1170-1176.

[6] R. Nevatia and K.R. Babu, "Linear Feature Extraction and Description," Computer Graphics and Image Processing, Vol. 13, 1980, pp. 257-269.

1.5 SHAPE MATCHING AND IMAGE SEGMENTATION USING PROBABILISTIC LABELING

Bir Bhanu

INTRODUCTION

In this contribution we present an introduction and summary of the work in the areas of shape matching and image segmentation using the technique of probabilistic labeling. Details about the work can be found in [1].

The problem of assigning names or labels to a set of units/objects is the key problem in computer vision, image analysis and pattern recognition. Since all the labels are not possible for a given unit, constraints based on contextual information, called the world model, are used to obtain a consistent and unambiguous valid assignment of the units. Local parallel processes are a very efficient way of assigning labels. The features of such algorithms include the propagation of local contextual information in a paradigm of competition and cooperation, locality and speed. In general the task of assigning names to units only on the basis of features of the units is very difficult since any segmentation based on low-level analysis is bound to contain errors and the computed features are

noisy. The solution to this problem is to delay any firm commitment until all the contextual information has been used. Depending upon the type of constraints embodying the world model, the problem can be attacked by discrete methods (discrete relaxation) or continuous methods (continuous relaxation, also called probabilistic or stochastic labeling). Recent surveys on these algorithms applied to low level vision and symbolic matching can be found in [2, 3]

Waltz [4] used the notion of constraints imposed by the world model for the description of scenes made up of complex polyhedra. Rosenfeld, Hummel and Zucker [5] proposed a parallel version of this algorithm. Barrow and Tenenbaum [6] and Rosenfeld et al. [5] introduced the idea of stochastic labeling. The nonlinear algorithm proposed in [5] has been extensively used in various applications [3]. Marr et al. [7] have used similar ideas. Theoretical analysis of convergence and stability properties of this algorithm have proven to be difficult as shown by Zucker et al. [8, 9]. Faugeras and Berthod [10 to 12] reformulated the stochastic labeling problem by explicitly maximizing a criterion function based on the inconsistency and ambiguity of classification. This criterion is maximized using a gradient projection method. A similar idea has been proposed by Ullman [13].

In this work we extend the stochastic labeling technique of Faugeras and Berthod [11] to do shape matching of two and three

dimensional objects in a hierarchical manner and to perform the segmentation of images. New results are presented [1] in the areas of shape matching of nonoccluded and occluded objects in two-dimensions, three-dimensional data acquisition, surface approximation by polygons, shape matching of three-dimensional objects and the segmentation of images having unimodal gray level distributions. The same stochastic labeling technique is used in both shape matching and segmentation with various extensions. The power of the techniques in 2-D is demonstrated by the examples taken from synthetic, aerial, industrial parts and microscope images where the matching is done after using the actual segmentation methods. In 3-D the complexity of the objects viewed is typified by a complicated casting of an automobile. In shape matching our concern will be with the shape properties of an object only. Other cues such as color and surface texture etc. have not been used. For the segmentation of images, aerial and biological gray level images have been considered. In the next section, we present an overview of the work and summary of contributions.

OVERVIEW AND SUMMARY OF CONTRIBUTIONS

The stochastic labeling technique developed by Faugeras and Berthod [11] is extended to do shape matching of two-dimensional objects in a hierarchical manner. The technique explicitly maximizes a criterion function based on the ambiguity and inconsistency of classification. Shape matching is viewed as a

segment matching problem, where a piece of a shape is recognized as an approximate match to a part of a larger shape. The hierarchical nature of the algorithm reduces the computation time and uses results at low levels to speed up and improve the accuracy of results obtained at higher levels. The two-dimensional shapes are represented by their polygonal approximation by finding the points of maximum curvature on their boundary. As compared to the previous studies in 2-D shape matching, the hierarchical stochastic labeling technique developed here, has been applied not only to the synthetic images, but also to the real images taken from the aerial, industrial parts and biological fields. The technique allows for the changes in scale, rotation, translation and significant changes in shape. The results of shape matching are used to compute the rotation. Various strategies that lead to faster computation are described. Although the technique is computationally more costly than the other feature based or correlation techniques, it is more robust and flexible. It allows for the parallel implementation in hardware. This method is also used when the the objects partially occlude, but our objective is to match only the object of interest.

The hierarchical stochastic labeling technique as described above is further extended to do shape matching of two-dimensional occluded objects. For each of the objects participating in the occlusion, there is a hierarchical process. These processes are executed in parallel and are coordinated in such a way that the

same segment of the apparent object, formed as a result of occlusion of two or more actual objects, is not matched to the segments of different actual objects. This problem is solved by combining the gradient projection method and penalty function approach. Results are presented on synthetic objects, a sequence of microscope images and industrial images when two or three objects partially occlude.

In three dimensional scene analysis, a method based on a laser triangulation principle to acquire 3-D data is described. The problems related with the 3-D data acquisition and geometric processing are addressed. Generation of 3-D object descriptions in terms of polygons which approximate the surface of an object has been studied. A new technique for the approximation of 3-D objects by a set of planar faces is proposed. This is used to generate a 3-D model of an object in terms of faces approximated by polygons. The technique is a sequential region growing algorithm. It is not applied to range images, but rather to a set of 3-D points. It is not directly related to how the 3-D surface points were acquired, but is tied to the sampling distances between the points on the object. It is not restricted to single view range data images, but is applicable to a composite object and does not require the ordering of points. It finds the convex faces of the object, but the information exists to merge convex parts of nonconvex faces. The 3-D model of an object is obtained by combining the object points from a sequence of range data images corresponding to various views of the

object, applying the necessary transformations and then approximating the surface by polygons. Such a representation should be of use not only in 3-D scene analysis, but also in computer graphics and reduction of terrain data obtained by synthetic aperture radar etc. Results are presented on a fairly complex industrial object.

The problem of shape matching of real world 3-D objects has been attacked by using the hierarchical stochastic labeling technique used in 2-D. In 2-D the matching is "segment matching," but now we have "face matching." A system for 3-D scene analysis is presented. This is useful for shape matching of real world 3-D objects. Using several examples of an automobile casting, it is shown that hierarchical stochastic labeling technique can be successfully used to accomplish partial shape recognotion in 3-D. The results of shape matching are used to compute the orientation of the object in space.

The segmentation of images having unimodal gray level distributions is also considered. Here we use the same stochastic labeling technique used for shape matching. Unimodal histograms are typically obtained when the image consists of a large background region with other small but significant regions. For example, in the biomedical area the extraction of the boundaries of various types of cells is complicated by the fact that cells are very close together, their boundaries are poorly defined and the gray level histogram is unimodal. Similarly

unimodal histograms are obtained for aerial pictures because the range of intensities of the many different objects overlap. The technique developed here provides control over the relaxation process by allowing the user to choose three parameters which can be tuned to obtain the desired segmentation results at a faster rate. This is the major advantage of the technique compared to the classical nonlinear relaxation method used in segmentation [14]. Aerial and biological cell examples are presented.

Finally we present several directions for future research in shape matching of two and three dimensional objects, segmentation of images and use of these concepts in the analysis of a real sequence of images [1].

REFERENCES

[1] B. Bhanu, "Shape Matching and Image Segmentation Using Stochastic Labeling," Image Processing Institute, USCIPI Report 1030, Univ. of Southern California, Los Angeles, CA.

[2] O.D. Faugeras, "Optimization Techniques in Image Analysis," Proc. of the 4th Int. Conf. on Analysis and Optimization of Systems, Lecture notes in Control and Information Sciences, Springer-Verlag, 1980, pp. 790-823.

[3] L. Davis and A. Rosenfeld, "Cooperative Processes for Low-Level Vision: A Survey," Dept. Comp. Sci., Univ. of

Texas, Austin, Tech. Rep. 123, Jan. 1980.

[4] D.L. Waltz, "Generating Semantic Descriptions from Drawings of Scenes and Shadows," MIT technical report AI271, Nov. 1972. For briefer versions see D. Waltz, "Understanding the Drawings of Scenes and Shadows, "in P.H. Winston, Ed., The Psychology of Computer Vision, McGraw Hill, New York, 1975, pp. 19-91.

[5] A. Rosenfeld, R. Hummel and S.W. Zucker, "Scene Labeling by Relaxation Operations," IEEE Trans. Systems, Man and Cybernetics, vol. SMC-6, pp. 420-433, 1976.

[6] H.G. Barrow and J.M. Tenenbaum, "MSYS: A System for Reasoning About Scenes," Tech. Note 121, Artificial Intelligence Center, SRI Intl., Menlo Park, CA, 1976.

[7]. D. Marr, T. Poggio and G. Palm, "Analysis of a Cooperative Stereo Algorithm," Biol. Cybernetics, Vol. 28, pp. 223-239, 1977.

[8] S.W. Zucker, and J.L. Mohammed, "Analysis of Probabilistic Relaxation Labeling Processes," Proc. IEEE Comp. Soc. Conf. on Pattern Recognition and Image Processing, Chicago, 1978, pp. 307-312.

[9] S.W. Zucker, E.V. Krishnamurthy and R.L. Haar, "Relaxation Processes for Scene Labeling: Convergence, Speed and Stability," IEEE Trans. on System, Man, and Cybernetics,

Vol. SMC-8, pp. 41-48, Jan. 1978.

[10] O.D. Faugeras and M. Berthod, "Improving Consistency and Reducing Ambiguity in Stochastic Labeling: An Optimiztion Approach," to appear in IEEE Trans. Pattern Analysis and Machine Intelligence.

[11] O.D. Faugeras and M. Berthod, "Using Context in the Global Recognition of a Set of Objects: An Optimization Approach," Proceedings of the 8th world computer congress (IFIP 80), Tokyo, pp. 695-698.

[12] O.D. Faugeras and M. Berthod, "Scene Labeling: An Optimization Approach," Proc. IEEE Comp. Sci. Conf. on Pattern Recognition and Image Processing, Aug. 6-8, 1979, pp. 318-326.

[13] S. Ullman, "Relaxation and Constrained Optimization by Local Processes," Computer Graphics and Image Processing , Vol. 10, pp. 115-125, 1979

[14] A. Rosenfeld, "Image Understanding Project Status Report," Proc. DARPA Image Understanding Workshop, April 1979, pp. 14-24.

1.6 CORNER DETECTION FOR FINDING BUILDINGS IN AERIAL IMAGES

A. Huertas

INTRODUCTION

Building structures can be detected using several approaches, two of which are: a) By obtaining a set of elementary features such as object boundaries and matching groups of them against specific object models stored in a visual memory (see [1] in this technical report), and b) By implementing a reasoning system that operates on a similar set of elementary features, knowing what it is looking for, and following the hypotheses formation/validation paradigm to provide consistent interpretations about the presence of building structures in aerial images.

One of such sets of elementary features is the subject of this section. In particular, the corners formed by long line segments obtained by a line finder that exhibits good corner detection characteristics, can be used to form building substructure hypotheses under a general set of assumptions.

Untrained human observers may have some difficulty locating

buildings in aerial images and therefore rely on familiar cues such as shadows, nearby roads, and apparent structure. But perhaps the most important feature are the observed geometric corners formed by building sides.

In this section we give a set of general assumptions regarding the perception and interpretation of corners derived from detected image features and a priori knowledge that could be available such as the direction of illumination.

CORNER DETECTION AND INTERPRETATION

A corner is defined at the intersection of two lines if the angle between them is approximately 90 degrees. For our purpose, lines forming approximately 180 degree corners are also extracted since otherwise straight lines can be broken for a variety of reasons.

When corners are extracted, several parameters can be associated with them for further processing. Some parameters include location, the characteristics of its components, the direction of the bisector, connectivity with other corners, and color (the brightness of the surrounded region). Other hypothesized parameters can also be associated with them, such as kind (object corner, shadow corner), compatibilities with other corners and type (concave, convex, straight).

Consider non occluded buildings (by other buildings, or clouds, or by shadows cast by other buildings) as having the appearance of box like objects. The boundaries of such boxes consist of straight sides forming 90 degree corners if the sides have only two possible orthogonal directions. Hence, regardless of direction of illumination, the following assumptions are made: Corners formed by long segments correspond to either object corners or to shadow corners in the image. Shadow corners are formed by pairs of line segments detected as a result of a shadow cast by an object corner. Shadows cast by buildings sides on the ground have a predictable width and therefore can be defined by pairs of dark anti-parallel segments [2].

Assuming a bright object against a dark background for the building shown in Fig. 1, the following interpretations are given: 1) A pair of line segments forming a corner around a bright region correspond either to an object convex corner (c1,c3,c4,c5) or to a shadow corner (c8) cast by an object concave corner (c5). If the bright corner is a shadow corner, there exists one, possibly two, dark anti-parallel segments paralle' to one or both of the shadow segments. If these dark apars lie between the shadow line segments and the projected position of the sun, then the bright corner is a shadow corner. Otherwise, the bright corner is an object convex corner. A similar interpretation can be derived for pairs of segments forming corners around dark regions.

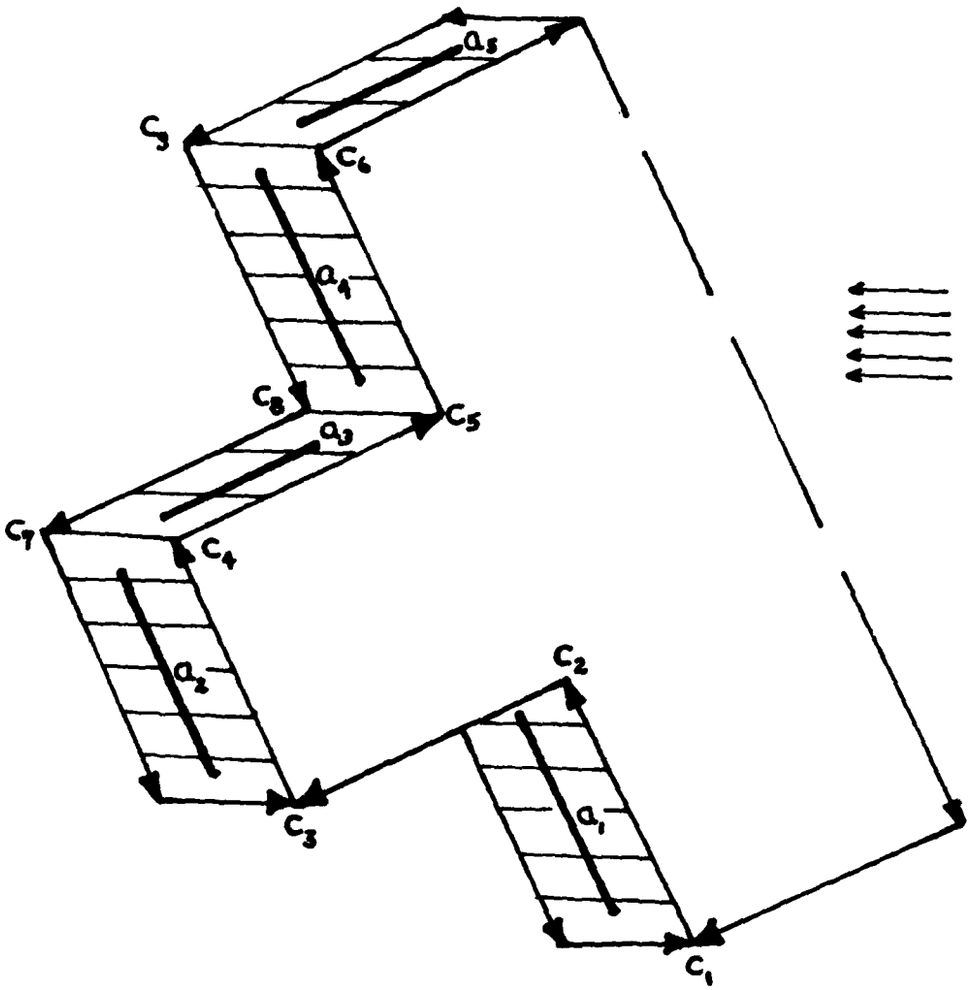


Fig. 1. Object and shadow corners.

In addition, or in the absence of shadow information, building corners must meet basic sets of geometric compatibility constraints. A single corner presupposes the existence of a compatible corner along their line segment components. Groups of corners can be taken together to satisfy even more complex compatibilities and consistent global interpretations can be given.

Figure 2 shows a detail of an aerial image containing buildings. The image was convolved with a Laplacian Gaussian filter for local edge detection [3]. Figure 3 shows the line segments obtained by a previously reported technique [2]. Figures 4 and 5 show all the corners extracted and those formed by long segments. Only corners whose components actually intersect were detected but suitable search windows can be defined for less well defined corners.

We are currently working in the definition of rules for hypotheses that could be derived from corner information, as well as the required compatibility functions among corners for the purpose of building detection in aerial images.

REFERENCES

[1] G. Medioni, "Matching of a Map with an Aerial Image," USCIPI Report 1050, Sept. 1981.

[2] R. Nevatia and K.R. Babu, "Linear Feature Extraction and

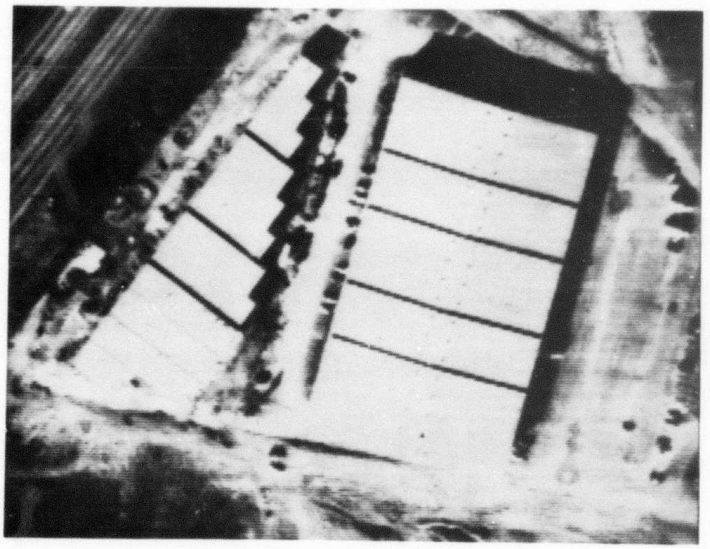


Fig. 2. BUILDINGS.

Fig. 3. BUILDINGS line segments.

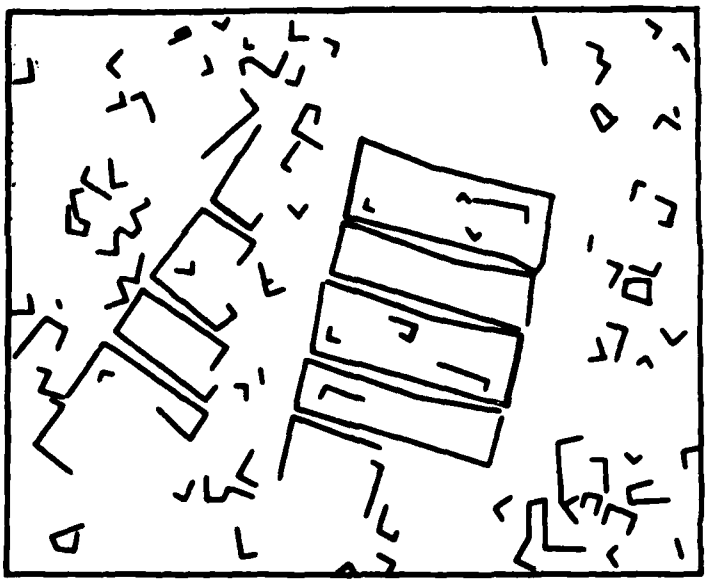


Fig. 4. CORNERS.

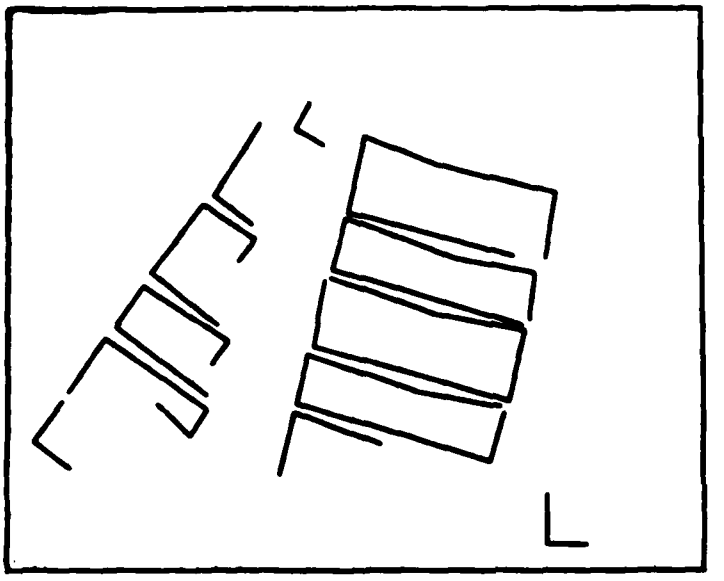


Fig. 5. CORNERS formed by long segments.

Description," Computer Graphics and Image Processing, Vol. 13, 1980, pp. 257-269.

[3] A. Huertas and R. Nevatia, "Edge Detection in Aerial Images Using Laplacian-Gaussian Filters," USCIPI Report 1010, April 1981, pp. 16-26.

1.7 STRUCTURAL TEXTURE ANALYSIS APPLICATIONS

F. Vilnrotter* and R. Nevatia

INTRODUCTION

In previous reports, we have described programs used to generate descriptions of natural textures [1-2] and extract and describe texture primitives [3] and the spatial relations between them [4-5]. Short descriptions of some of these programs may also be found in [6-7]. These techniques are useful for generating structural texture descriptions. We have applied them to the tasks of texture recognition, and surface orientation determination (the latter application is only partially implemented). This section describes these applications; a complete report on this work including the texture description programs will be available separately [8].

Next, the problem of determining the orientation of surface, using the gradient of the texture is discussed. Work done by Bajcsy [9] and Stevens [10], [11] in this area is briefly discussed. An orientation analysis scheme is suggested which is

---

*F. Vilnrotter was supported by a Hughes Aircraft Company Doctoral Fellowship.

based on these methods and incorporates some of the texture analysis techniques presented earlier. Preliminary results for a number of texture images exhibiting texture gradients are discussed.

TEXTURE RECOGNITION

A texture recognition algorithm that uses the descriptions generated by our texture analysis programs is discussed below. The descriptions consist of the periodicity of the texture and the size and shape of the texture elements. Details of descriptions are given in part in [1-3,6] and all details may be found in [8]. Hopefully, the discussion of the recognition algorithm below is self-explanatory in terms of the descriptions used. The recognition scheme is basically a decision tree. Eleven types of textures were used in our experiments.

Texture Recognition Algorithm

The structure of the algorithm used for texture classification is shown in Fig. 1. The texture classified are: floor grating (dark dot pattern), brick wall, aerial view of city, raffia (woven palm), herringbone material, wood grain, aerial view of water, straw, grass, sand, and wool.

Most of the textures used are from the Brodatz album [12]. The exceptions are the floor grating, brick wall, and aerial city patterns. Aerial city pictures taken at different orientations and different scales were used. Only 11 samples are found. All

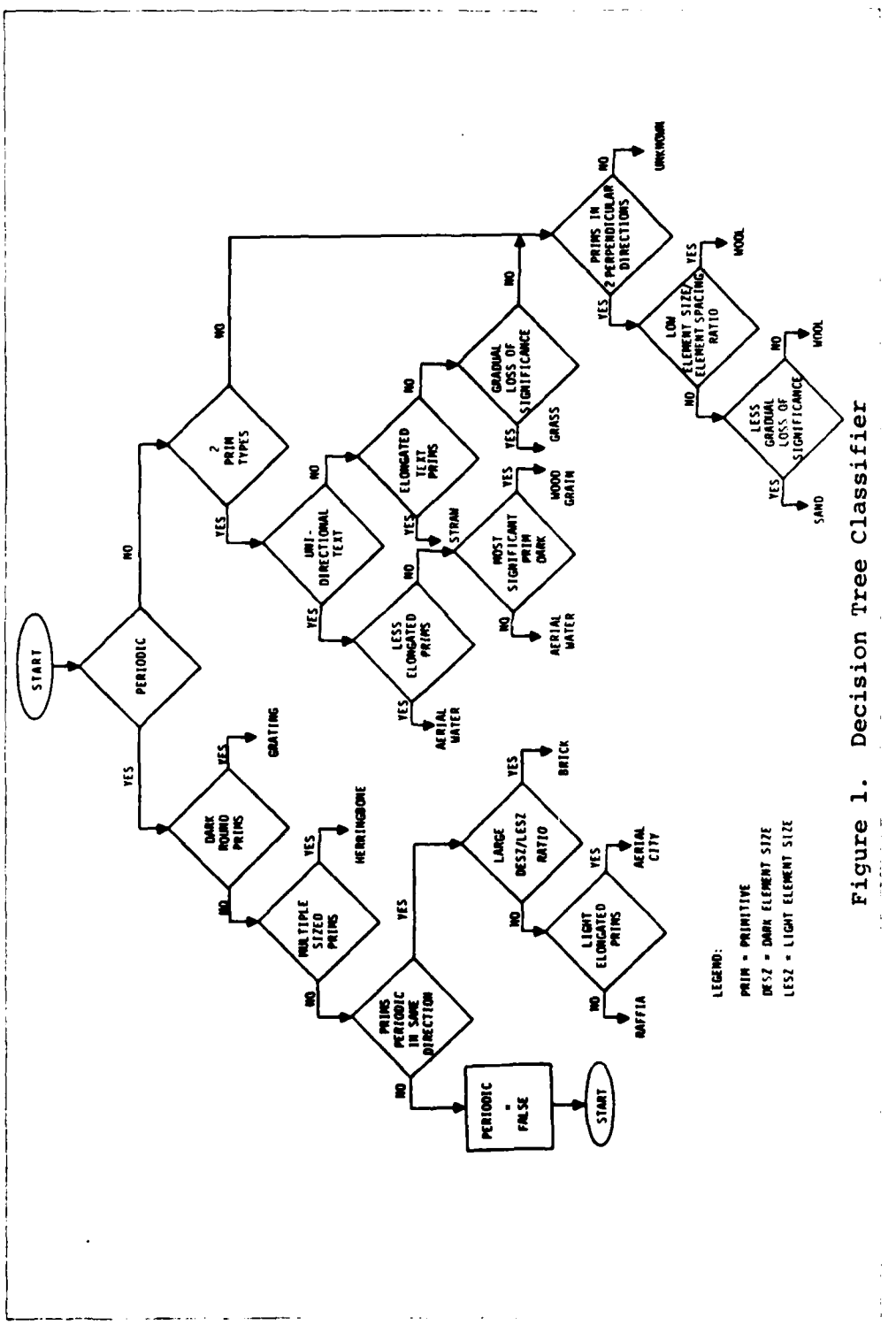


Figure 1. Decision Tree Classifier

other texture groups consist of 16 samples each. Pictures of both shifted and unshifted brick patterns are used.

The decision tree form was chosen for this texture classification scheme. The structure of the tree allows the classification information to be weighted by means of relative ordering. For example, the test for periodicity is encountered at an earlier stage than any of the aspect ratio tests. Clearly texture period information is being given more significance than aspect ratio information according to this classification scheme.

The first texture characteristic considered is periodicity. If a texture is non-periodic it will be classified by the subtree to the right of the root node. However, if the texture exhibits signs of periodicity but fails the tests for each of the 5 regular texture patterns it is sent back to the root node, re-labeled as a non-periodic texture. Due to this loop in the structure, the classification algorithm does not have the exact form of a binary decision tree. However, for convenience tree terminology will be used in the discussion. The loop is introduced to accommodate textures which are basically non-periodic, but may show evidence of periodicity some of the time. The wood grain, water and straw textures exhibit this characteristic. The opposite is not as likely to occur, i.e., the periodic textures are not mistakenly sent down the non-periodic branch.

No absolute texture element dimensions or intensity values are used; and all directional information is relative as well. In this way, the analysis should be insensitive to scaling, rotation and degree of contrast within the image. Measures which are used are primitive eccentricity, dimension to period ratio, the number and significance of texture element types, and relative intensities and orientations of texture primitive types. The details of each decision box for the two main decision branches is given below:

Periodic Branch

1) Dark Round Primitives - More than four dark primitives are merged into one primitive type.

2) Multiple Sized Primitives - Multiple sized primitives are found in two perpendicular directions. The texture period is equal to the sum of the element sizes. The sum of the (size/period) ratios for the four most significant primitives is less than .2.

3) Most Significant Primitives are Periodic in the Same Direction - The two most significant primitives exhibit an element spacing value for the same scan direction.

4) Large (Dark Element Size/Light Element Size) Ratio - The dark primitive is at least twice as wide as the light primitive in the most significant scan direction.

5) Light Elongated Primitives - The ratio of the dimension in the most significant scan direction to the dimension in the direction perpendicular to the direction of scan is less than .4.

Non-Periodic Branch

1) Uni-Directional Texture - The two most significant primitives are found in the same scan direction. The significance numbers associated with these two primitive types are greater than all other significance numbers by at least .66 (out of a possible 1.0).

2) Less Elongated Primitives - The sum of the aspect ratios for the two most significant primitives is at least as great as .175.

3) Most Significant Primitive is Relatively Dark.

4) Elongated Texture Primitives - The minimum aspect ratio of the four most significant texture primitives is no larger than .18.

5) Two Primitives Types - Primitives are found for the two most significant (intensity,direction) pairs.

6) Gradual Loss of Significance - There is no abrupt loss of significance after the fourth (intensity,direction) pair. The difference is smaller than .3.

7) Primitives Found in Two Perpendicular Directions.

8) Low Size/Spacing Ratio - There is a low (element size/element spacing) ratio for relatively dark primitives in two perpendicular directions. The sum of both ratios is less than .53.

9) Less Gradual Loss of Significance - The loss of significance after the fourth (intensity,direction) pair is at least .3.

No attempt has been made to optimize this decision scheme. The classification results are discussed in the following section.

Classification Results

Classification results are given in the confusion matrix shown in Table 1. The types of samples to be classified are listed to the left of the matrix. Each row shows how a specific set of samples was classified. For example, 15 aerial water texture samples were correctly classified, while one sample was incorrectly classified as wood grain. One hundred seventy-one samples were classified in all. In most cases the samples came from 512 x 512 pixel texture images. These were divided into sixteen 128 x 128 pixel non-overlapping texture subwindows. However, in the case of the aerial city samples only 11 samples were available. These were cropped from two different satellite images of the San Francisco area. The 16 brick wall samples were taken from three separate brick wall images. One hundred

Table 1. Confusion Matrix

Type	Floor Grating	Herring-Bone	Raffia	Brick	Aerial City	Aerial Water	Wood Grain	Straw	Grass	Sand	Wool
Floor Grating	16	0	0	0	0	0	0	0	0	0	0
Herring-Bone	0	16	0	0	0	0	0	0	0	0	0
Raffia	0	0	16	0	0	0	0	0	0	0	0
Brick	0	0	0	16	0	0	0	0	0	0	0
Aerial City	0	0	0	0	11	0	0	0	0	0	0
Aerial Water	0	0	0	0	0	15	1	0	0	0	0
Wood Grain	0	0	0	0	0	2	14	0	0	0	0
Straw	0	0	0	0	0	0	0	16	0	0	0
Grass	0	0	0	0	0	0	0	0	11	5	0
Sand	0	0	0	0	0	0	0	0	0	14	2
Wool	0	0	0	0	0	0	0	0	2	3	11

fifty-six samples were correctly classified to give an overall success rate of 91.23%. It should be noted that additional contextual information, e.g., color, scale, and type of scene would probably have improved the results obtained. However, no information of this type was used in the classification scheme in order that the strength of the texture descriptions used could be tested in isolation.

There are no mismatches for the highly structured, regular texture group. However, there are a number of non-periodic texture samples which are classified incorrectly. One source of confusion is between the water and wood grain textures. Both are one-dimensional textures made up of elongated texture primitives. The wood grain primitives tend to be more elongated than the water wave primitives. There is little else which is noticeably different from a structural point of view. Hence, confusion of these two texture types is predictable. Additional contextual information, e.g., the scale information for both textures, would improve the classification results.

Another area for confusion is the set of textures consisting of (grass, sand, and wool) exhibit the least amount of structure of the entire set. The edge images, ERAs, ERA descriptions, composite texture primitive masks, and texture primitive descriptions are very similar for all three types. This is because the program is not designed to measure the types of features which most readily differentiate these textures. In

light of these description similarities, confusion among members of this group is to be expected. It should be noted that none of these texture samples were confused with textures from outside of the group and none of the other texture samples were mistakenly classified as one of these. The same can be said of the subgroup formed by the water and wood grain textures.

In summary 156 samples out of 171 were correctly classified to give an overall success rate of 91.23% for this classification scheme. These results are extremely encouraging. It would seem that the information extracted by the algorithms presented earlier in this thesis describe meaningful texture characteristics.

SURFACE ORIENTATION ANALYSIS

In Figs. 2 and 3 two texture gradient images are shown. Figure 2 is an image of a brick wall and Fig. 3 is an image of a redwood shake roof. In each case the textured surface is at a non - zero angle with respect to the image plane. The brick wall recedes to the left of the image, while the shake roof slants away from the viewer toward the top of the image.

One cue which we use to infer information about the orientation of textured surfaces is the texture gradient, i.e., the relative change in size or period of the elements making up the textured region within the image. Assuming that the textured surface being viewed is homogeneous, the element sizes of like

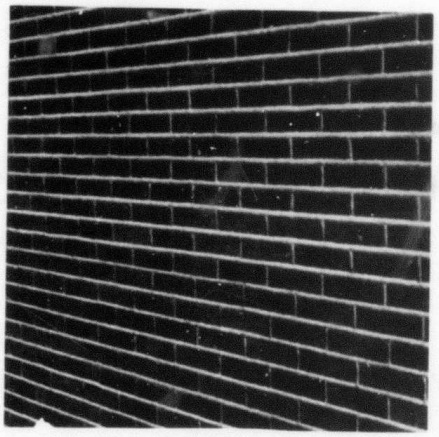


Figure 2. Brick Wall

Figure 3. Redwood Shake Roof

texture primitives should decrease with increased distance from the viewer. The direction of maximum rate of change of depth with respect of the observer should be discernible as well as the degree of surface slant in this direction. A convenient way to represent these two surface characteristics is via the gradient space.

Let $z = f(x,y)$ be a function defining a planar surface in 3-Space, where the image plane is parallel to the x-y plane, (see Fig. 4). The surface normal, N, can be defined as follows: $N = (f_x, f_y, -1)$. Letting $p = f_x$, and $q = f_y$, we have the gradient vector, $G = (p,q)$. The direction of G is $TAN^{-1}(q/p)$, while the magnitude of G is $SQRT(p^2 + q^2)$. The direction of G denotes the direction of the greatest rate of change within the image, while the magnitude of G determines its quantity. Also, the tangent of the angle that the surface makes with the x-y (or image) plane is equal to the magnitude of G. Therefore, the orientation of a surface in 3-Space can be represented as a point in the gradient space.

In [10] Stevens suggests an alternate set of coordinates to represent surface orientation. He suggests the pair (σ,τ), where

$$\sigma = \tan^{-1}((p^2+q^2)^{1/2}),$$

and

$$\tau = \tan^{-1}(q/p).$$

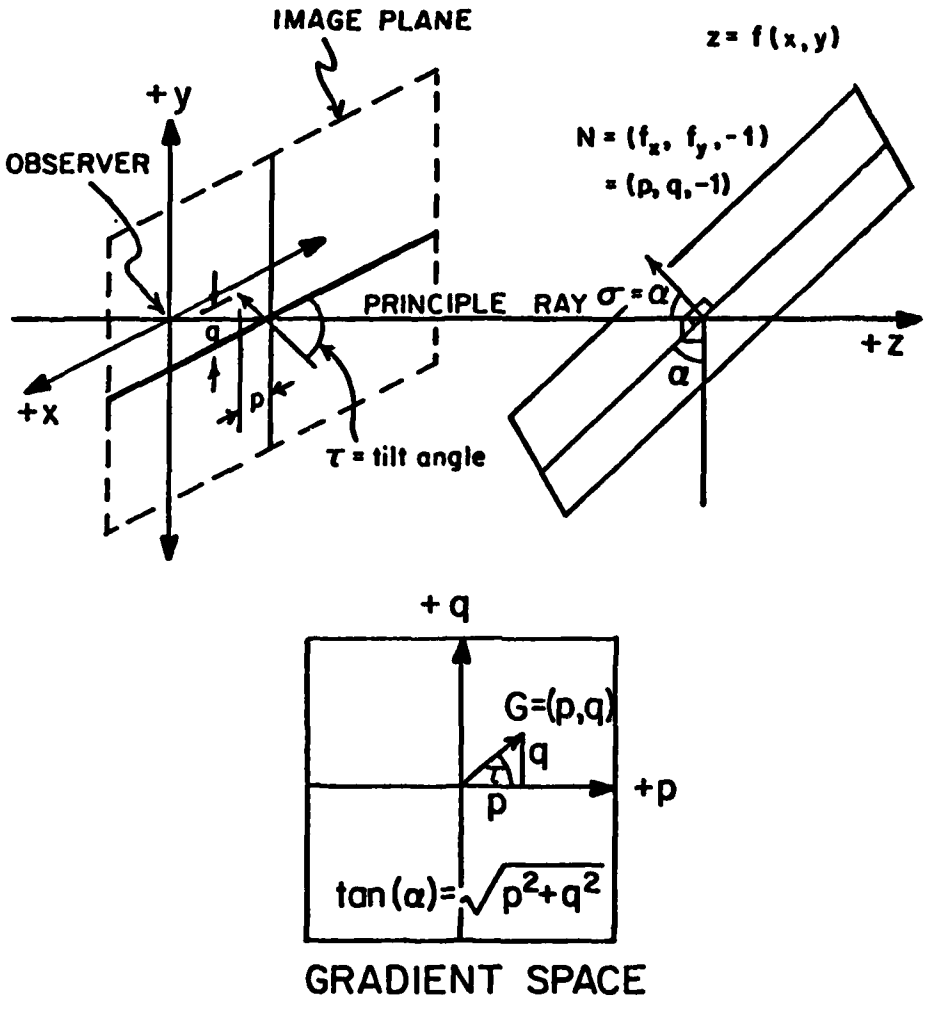


Figure 4. Gradient Space Scheme

Stevens' slant and tilt angle terminology will be used. The tilt angle, τ , is an angle made with the horizontal axis of the image plane. It is the projection of the surface normal onto the image plane. Stevens has shown that this direction coincides with the direction which exhibits the greatest rate of change of distance from the observer to the surface. This is precisely the direction of the texture gradient. Therefore, by calculating the texture gradient one can determine the tilt angle. The slant angle, σ, is the angle which defines how much the image plane orientation differs from the surface plane orientation. For a discussion of various forms of surface orientation representation see the works of Stevens [10] and Kender [13].

Calculating the tilt angle is fairly straightforward. It entails determining the direction of the gradient of any texture measure which is scaled (due to distance), foreshortened (due to surface orientation), or both scaled and foreshortened.

Determining the slant angle, σ , is more involved. One method suggested is to determine which texture measure corresponds to the characteristic dimension. That is, which texture measure is scaled but not foreshortened. The normalized gradient of this dimension, taken in the direction of the texture gradient, is equal to the tangent of the slant angle.

$$\frac{\nabla d}{d} = \tan \sigma, \qquad (1)$$

where d is the characteristic dimension. Characteristic dimensions are parallel to the image plane and are perpendicular to the local surface tilt. Therefore, after the surface tilt has been determined the orientation of the characteristic dimension is known. This scheme for calculating s cannot be used if the image is an orthographic projection, or if the elements exhibit successive occlusion. Alternative schemes are explored in [10] for handling these problems. Here it will be assumed that neither problem exists.

In [9] Bajcsy presents a method for calculating the angle formed by the image and surface planes. However, the dimension used in this case is the dimension oriented in the gradient direction. Therefore, it is both foreshortened and scaled.

Using the principles of projective geometry, the trigonometric rules pertaining to similar triangles and some small angle approximations, Bajcsy derives an expression for α, the angle formed by the surface and image planes.

$$\frac{-\tan\alpha}{\text{Focal Distance}} = \frac{\text{Fractional Change in Element Size}}{\text{Baseline in Image}} \qquad (2)$$

where the fractional change in element size and the baseline within the image are both measured in the direction of the texture gradient. Details of the derivation can be found in [8].

Both methods assume proximity to the line of sight defining the local image plane. All of the examples used in the following

section have maximum off center angle less than 10 degrees. Further work is necessary to define the transformation needed to correct for large off center angular separations. This transformation should take into account the image plane and lens system characteristics as well as the geometry needed for coordinate transformations.

A General Orientation Analysis Technique

Application of our structural texture analysis techniques to surface orientation analysis problem is in a preliminary stage. The process has not been fully defined or automated. An outline of a proposed algorithm is discussed below, and some preliminary results are presented in the following section. Although only part of the program is operational, see 2 below, most of the remaining program sections are defined and seem feasible from a programming point of view.

One effect which must be anticipated by this technique is the possible changing orientation of the texture primitives. If the texture gradient was strong enough the height of the brick primitives in Fig. 2 would be found in the 120 degree scan direction in the lower left hand part of the image and in the vertical scan direction in the right and central areas. One possible solution is discussed in (3) below.

A possible scenario for detecting surface orientation is as follows:

1. Divide the textured region into locally uniform subwindows, i.e., subwindows which exhibit very little, if any, element size variation. This algorithm is not yet defined. However, as a first approximation the image can be divided into subwindows which accommodate the largest texture elements. Then element sizes can be averaged over each texture subwindow to produce an element size for that location in the image. This was done manually for the two examples discussed below. (In order to automatically determine the largest element size, one might calculate modified ERAs (see 2 below) for a large range of distances, say up to one third of the largest image dimension, over the entire texture image. Then an ERA interpretation routine, similar to the one presented in [2], can be used to determine the largest texture element dimensions.)

2. Calculate modified ERAs for each subwindow within the region. Only the first match encountered will be recorded for a particular directional scan. Hence, no element size or spacing repetitions should be counted. It is hoped that this will prevent repetitions from being interpreted as element size variations within a given subwindow. The modified ERA calculation scheme is operational and has been used to produce the results shown in Fig. 5.

3. Look for a dimension exhibiting strong results for each subwindow. A dimension, d, which is locally uniform but

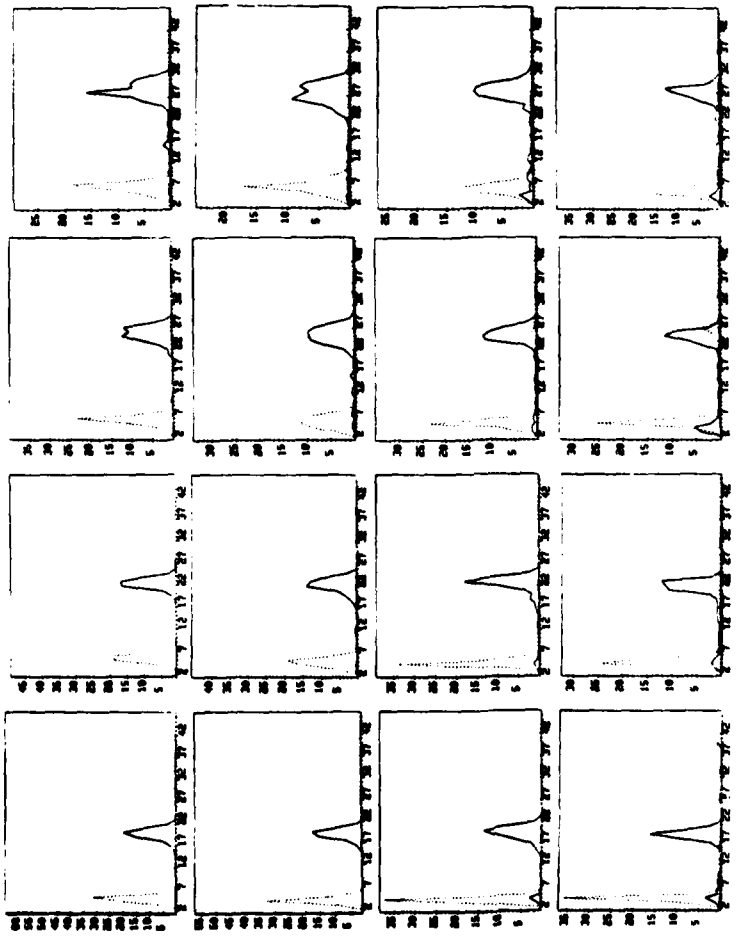


Figure 5. Vertical Brick Element Size Graphs for Example 1.

which reflects the gradient of the texture is sought. It should be verified that all of the ERA results chosen refer to the same texture primitive type. One way to achieve this end is to extract texture primitives for overlapping subwindows and verify that enough of the primitives extracted belong to both subwindow neighbors. In this way textural elements which shift orientation due to the effect of angular perspective can be found in each subwindow.

4. Create a matrix, M, of the centroid values of the element size peaks for d. Use a gradient operator on M to calculate the value of the tilt, or gradient, angle of the surface within the image.

5. Knowing the tilt angle means that the orientation of the characteristic dimension is also known. It can then be ascertained if either the characteristic dimension or the gradient dimension is already available as part of the set of ERA results. If this is the case then the rest of this step can be skipped. If this is not true then the characteristic or gradient dimension of some non-background texture primitive type must be measured from texture primitive masks. (These measurements should proceed outward from the e emental centers of mass.) Either of these two dimensions can be used for the slant angle calculation. Histograms of these dimensions should be kept so that the final calculations can be made using the highest amplitude/lowest standard deviation

results. When measuring the characteristic dimension the original set of subwindows can be used. However, when calculating the gradient dimension the subwindows might have to be recropped to provide the correct center to center angle.

6. At this point either Eq. 1 or 2 can be used to calculate the surface slant angle to complete the procedure. If the characteristic dimension is known then Eq. 1 would be used. If the dimension oriented in the texture gradient direction is known then Eq. 2 would be used. Both slant angle calculation methods are discussed above.

Texture Gradient Examples

In this section two texture gradient examples are presented and discussed. They make use of the general method outlined in above.

Example 1

Consider the brick wall image in Fig. 2. This image is 512 x 512 pixels. It was divided into 16 128 x 128 pixel subimages, and ERAs were calculated for each of these. The vertical element size ERAs for each subwindow are shown in Fig. 5. As expected the element sizes decrease toward the left side of the image. Figure 6(a) shows the brick vertical element size dimension matrix. In Fig. 6(b), the results of the gradient calculation are shown; and Fig. 6(c) shows the results of the

19.759	22.547	25.644	28.551
19.515	22.406	25.176	27.450
19.368	22.496	25.306	28.182
18.938	21.868	24.965	28.029

(a) Brick Vertical Dimension Matrix. Each value, d, is a dark, vertical element size.

$S_x = -23.146$ (1) $S_y = -.83$

$S_x = 21.778$ (2) $S_y = -1.094$

$S_x = -23.566$ (3) $S_y = -1.864$

$S_x = -22.577$ (4) $S_y = -.381$

(b) Gradient Results for (a).

$$\tau = \tan^{-1}\left(\frac{S_y}{S_x}\right) \qquad \sigma = \tan^{-1}\left(\frac{\sqrt{S_x^2+S_y^2}}{d}\right)$$

2.05°	2.88°
4.52°	$.967^\circ$

43.95°	40.90°
46.42°	41.74°

(c) Tilt and Slant Angle Matrices.

Figure 6. Analysis for Example 1.

tilt and slant angle calculations. In this case the vertical brick dimension is the characteristic dimension, hence, the method developed by Stevens will be used to calculate the surface slant. The slant and tilt calculations are carried out for the 4 interior subwindows of the image (Fig. 2). The tilt angle measured from the image is approximately 3^{o}. The tilt angle results range from $.967^{o}$ to 4.52^{o}. Unfortunately, the actual slant angle is not known for this image. However, the results found for this angle seem to be reasonable. The angle made by the image and surface planes, or similarly, the angle between the principle ray and the surface normal seem to be in the neighborhood of 45^{o}. In the next example the approximate angle formed by the surface and the image plane is known.

Example 2

Consider the shake roof image in Fig. 3. This image is 512 x 512 pixels. It was divided into 9 170 x 170 pixel subimages. (The last two rows and columns were not used.) ERAs were calculated for each of the nine subimages. The vertical element size ERAs for each subwindow are shown in Fig. 7. As expected the element sizes decrease toward the top of the image. Figure 8(a) is the matrix of the centers of mass for the vertical element size ERAs of Fig. 7. In Fig. 8(b) the results of the gradient and tilt angle calculations are shown, and Fig. 8(c) shows the results of the slant angle calculation for three image locations. The gradient direction dimension, i.e., the vertical

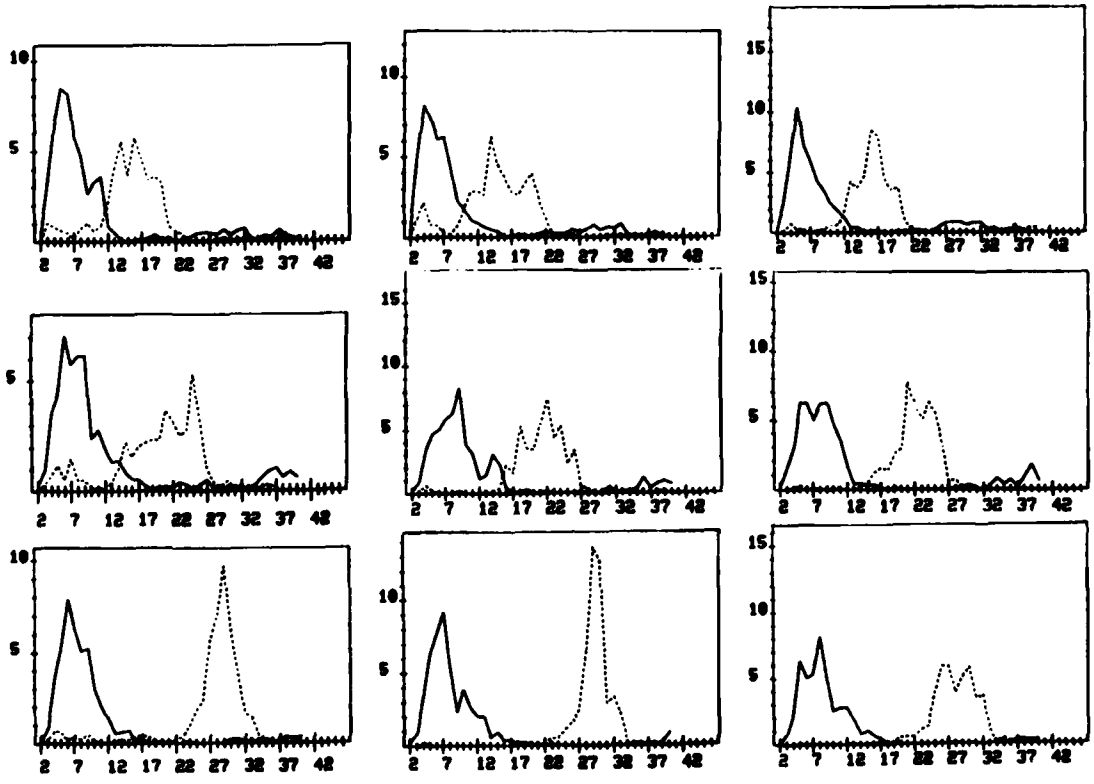


Figure 7. Vertical Brick Element Size Graphs for Example 2.

	1	2	3	
d_1	16.029	15.855	16.286	Image Baseline (340 pixels)
d_2	21.223	21.317	22.104	
d_3	28.808	29.347	28.146	

(a) Redwood Shake Vertical Dimension Matrix. Each value, d, is a light vertical element size.

$$S_x = 1.357 \qquad \tau = \tan^{-1}\left(\frac{S_y}{S_x}\right)$$

$$S_y = 51.623 \qquad \tau = 88.49^{\circ}$$

(b) Gradient and Tilt Angle Calculation.

$$\alpha = \tan^{-1}\left(\frac{(d_3-d_1)^* \text{ Focal Distance}}{\frac{1}{2}(d_3+d_1)^* \text{ Image Baseline}}\right)$$

Focal Distance = 50 mm = 2008.33 pixels

α_1	α_2	α_3
73.46°	74.17°	72.41°

(c) Slant Angle Result Matrix.

Figure 8. Analysis for Example 2

dimension, of the wood shake elements was already known via ERA calculation. Therefore, the the surface slant angle was calculated using the method developed by Bajcsy. (It would be very difficult to use the characteristic dimension in this example since the widths of the wood shake are variable.) The slant angle computations are carried out for 3 sets of data. The tilt angle is calculated once since there are only enough windows for one gradient computation. The tilt angle for the image (Fig. 3) is approximately 90^{o}. The tilt angle was computed to be 88.49^{o}. The slant angle was calculated to be approximately 71.97^{o}. (See Fig. 9.) The slant angle calculation results range from 72.405^{o} to 74.167^{o}.

Determination of window size is a problem which must be addressed. One possible solution is to calculate a set of ERAs for the entire image, initially, for a wide range of distances. The maximum element sizes found would then dictate the appropriate window size.

SUMMARY AND CONCLUSIONS

Structural texture analysis techniques previously developed were applied to 2 texture analysis problems, namely, texture recognition and surface orientation determination. The results obtained in both cases appear to be very promising.

First, a classification scheme using both one-dimensional texture descriptions and texture primitive descriptions was

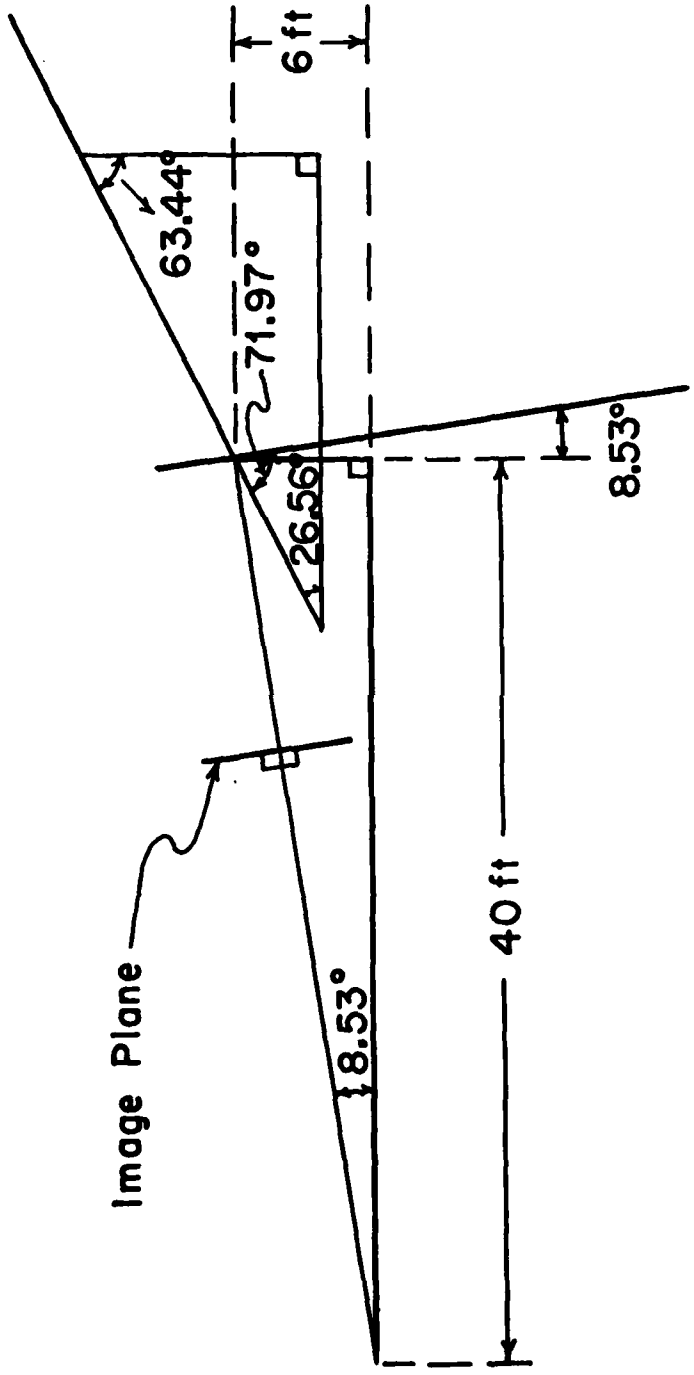


Figure 9. Geometry for Surface in Fig. 3.

presented and classification results were discussed. The algorithm was designed to classify 12 different types of textures, including 6 random and 5 periodic textures. The random textures were taken from the Brodatz album [12]. These are grass, sand, wool, water, wood, and straw. The 5 periodic textures came from a variety of sources ranging from aerial imagery to pictures taken by the author. They are raffia, herringbone material, floor grating, aerial city, and brick wall. The algorithm achieved an overall success rate of 91.23%. The classification scheme worked well for the non-periodic texture group, and extremely well for the highly structured regular textures, achieving 100% correct classification for this group. In those cases where there was confusion additional contextual information would have been helpful. For example, knowing scale information would have aided in distinguishing wood grain from the aerial water texture. As might be expected the amount of algorithm success varied directly with the amount of structure present in the texture. Since the range of textures handled by this algorithm is varied and any confusion encountered is restricted to small groups, (2-3) of similar textures it is fair to say that the description information extracted thus far corresponds to meaning textural features.

The surface orientation determination scheme presented is in preliminary form. The method described is only partly automated. It uses schemes developed by Bajcsy [9] and Stevens [10]. It also utilizes the techniques discussed in earlier reports. Some

preliminary results are presented and discussed. Tilt and slant angles are calculated for two images exhibiting non-zero texture gradients. The results for all known quantities are accurate to within 5 degrees. These results appear to be promising. However, more work needs to be done to completely automate the process.

REFERENCES

[1] R. Nevatia , K. Price and F. Vilnrotter, "Describing Natural Textures," USCIPI Semiannual Technical Report 860 March 1979, pp. 29-54.

[2] F. Vilnrotter, R. Nevatia and K. Price, "Automation Generation of Natural Texture Descriptions," USCIPI Semiannual Technical Report 910, September 1979, pp. 31-63.

[3] F. Vilnrotter, R. Nevatia, and K. Price, "Extraction of Texture Primitives," USCIPI Semiannual Technical Report 960, March 1980, pp. 48-59.

[4] F. Vilnrotter, R. Nevatia, and K. Price, "Determining Spatial Relationships Between Texture Primitives in Homogeneous Regular Textures," USCIPI Semiannual Technical Report 990, September 1980, pp. 10-26.

[5] F. Vilnrotter, R. Nevatia, and K. Price, "Automatic Grid Relation Extraction and Texture Reconstruction for Homogeneous Regular Textures," USCIPI Semiannual Technical

Report 1010, March 1981, pp. 27-42.

[6] R. Nevatia, K. Price, and F. Vilnrotter, "Describing Natural Textures," Proc. of the Sixth IJCAI-79, Tokyo, Japan, August 1979.

[7] F. Vilnrotter, R. Nevatia, and K. Price, "Structural Description of Natural Textures," Proc. of the Fifth IJCPR-80, Miami, Florida, December 1980.

[8] F. Vilnrotter, "Structural Analysis of Natural Textures," USC Computer Science Ph.D. Dissertation, to be published in 1981.

[9] R. Bajcsy, "Computer Identification of Textured Visual Scenes," Computer Science Report, Stanford University, 1972.

[10] K.A. Stevens, "Analysis and Representation of Visual Surface Orientation," Ph.D. Dissertation, MIT, 1978.

[11] K.A. Stevens, "Representing and Analyzing Surface Orientation," Artificial Intelligence: An MIT Perspective, pp. 104-125, Cambridge: MIT Press, 1979.

[12] P. Brodatz, Texture: A Photographic Album for Artists and Designer. New York: Dover, 1979.

[13] J.R. Kender, "Shape from Texture," Ph.D. Dissertation, Carnegie-Mellon University, 1980.

1.8 TEXTURE SYNTHESIS USING A PIECEWISE-LINEAR MODEL

D.D. Garber and A.A. Sawchuk

INTRODUCTION

Work presented earlier by Garber and Sawchuk [1-4] has shown that a carefully chosen linear model can be used to simulate a variety of natural textures. As an extension of first-order linear model a piecewise-linear model is proposed in this paper. Previous limited studies indicated that the slight improvement in simulation quality did not justify the added computational expense required to estimate parameters of the piecewise-linear model. Nevertheless, additional texture simulations using this method were done on a variety of textures and the results are presented in this paper. These results indicate that a single linear model produces texture simulations superior to those generated using a piecewise-linear model.

SINGLE LINEAR MODEL

A method of texture synthesis based on the linear model was presented by Garber and Sawchuk in earlier studies [1-4]. The model used is most simply stated as

$$V_{N+1} = \beta_i V_1 + \beta_2 V_2 + \ldots + \beta_N V_N + \beta_0 + \varepsilon. \quad (1)$$

Each individual pixel of a synthesized texture is computed as a linear combination of previously generated pixel values plus a noise term according to Eq. (1). The pixels are generated one-at-a-time along each row, row by row, until the entire synthesis image space is filled. Each pixel is a linear combination of only those pixels above it and thus, the synthesis process is causal.

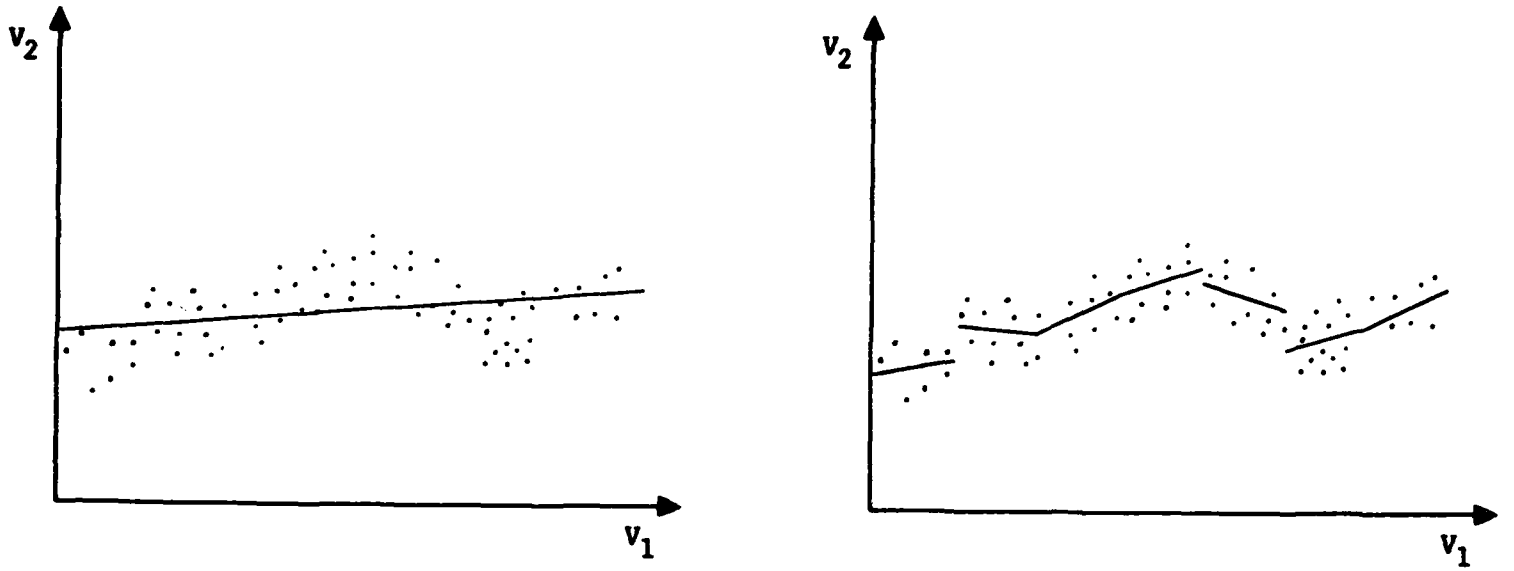
A least-squares method is used to estimate the β_i coefficients of the linear equation by fitting the equation to the sample texture data. Selection of the spatial relationship of the V_i's of the generation kernel is accomplished using ideas commonly employed in stepwise linear regression studies. This process is detailed in [1]. The variance of the random noise, ε, is chosen to be equal to the unexplained variance when the model is applied to the original sample parent texture data. The distribution of the noise used in this paper was Gaussian. Synthesis results using this single linear-model method are shown in Figs. 2(b) to 8(b). The original parent textures are shown in Figs. 2(a) to 8(a). The synthesis results are relatively good, especially when the vast data reduction is considered. The number of terms in each of the models used was less than 70.

PIECEWISE-LINEAR MODELS

When generating textures using the general linear model described by Eq. (1), the same model is used regardless of the values of the pixels $V_1,\ldots,V_N$. By developing more than one linear model and allowing the choice of the model at each pixel generation step in the synthesis process to be dependent on some functional value of $V_1,\ldots,V_N$, $F(V_1,\ldots,V_N)$ a new synthesis model is formed.

To illustrate this concept consider the data in Fig. 1(a). If we were to fit one linear model to the data in order to predict V_2 from V_1 it would look like the single line running through the data in Fig. 1(a). This linear model could then be used to predict V_2 based on the value of V_1. But if we allow the choice of our linear model to be dependent on the value of V_0, then for an incoming value of V_1 we choose a model whose domain includes V_1 to predict V_2. For 6 linear models, the straight lines are shown in Fig.1(b). The fit to the data using multiple linear models will always be as good as or better than that of the single linear model. That is, the mean square error will generally be reduced using multiple models.

Using multiple linear models for texture synthesis we would generate pixels V_{N+1} based on pixels $V_1,\ldots,V_N$ in the following way. First, we compute a function, F, of the $V_1,\ldots,V_N$ pixels



(a) Single Linear Model

(b) Piecewise-Linear Model

Fig. 1. Linear models.

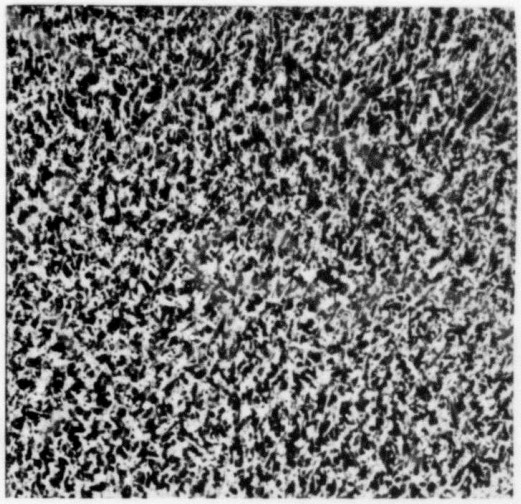
which allows us to choose the proper linear model. Then, using this model with the values $V_1,\ldots,V_N$, we predict V_{N+1} and add noise.

Ideally, the function F should be chosen to minimize the total mean square error resulting from fitting the limited number of models to the sample data. This is very difficult to do in practice however as for N larger than 3 we are fitting multiple hyperplanes to data in an N+1 dimensional space.

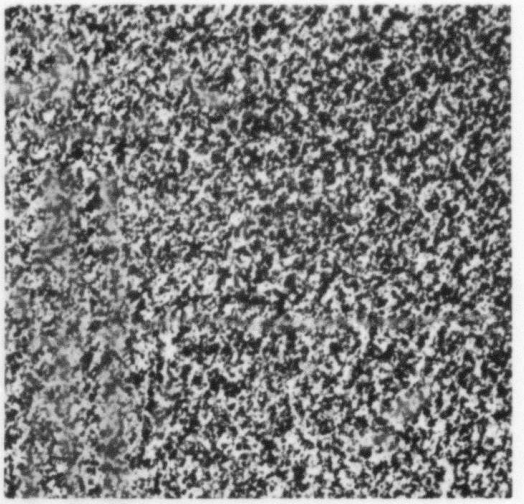
Texture synthesis results using the piecewise-linear model are shown in Figs. 2(c) to 8(c). In these cases, eight models were used and the model number was chosen by examining the pixel immediately to the left of the pixel being generated. The range of this pixel, 0 to 255, was divided into 8 equal subranges and the model was chosen according to the subrange into which the value fell. For each piece of the piecewise-linear model, a unique set of V_i pixels and β_i coefficients was chosen using the stepwise approach employed in model selection for the single linear model.

CONCLUSION

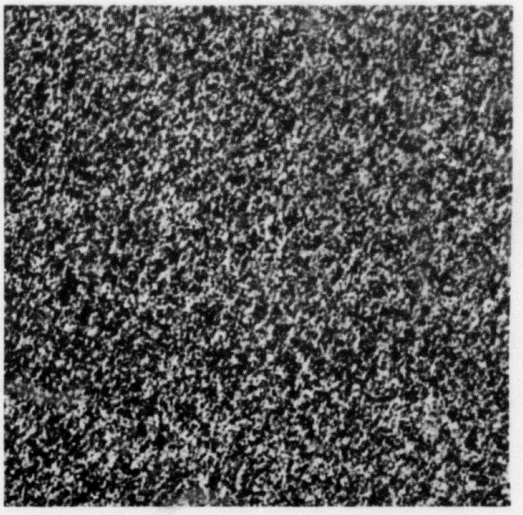
Generally speaking, the synthesis results using the piecewise-linear model approach are less successful than those obtained using a single linear model. At first, this seems to contradict the simple analysis indicated in Fig. 1. However,



(a) Original parent texture

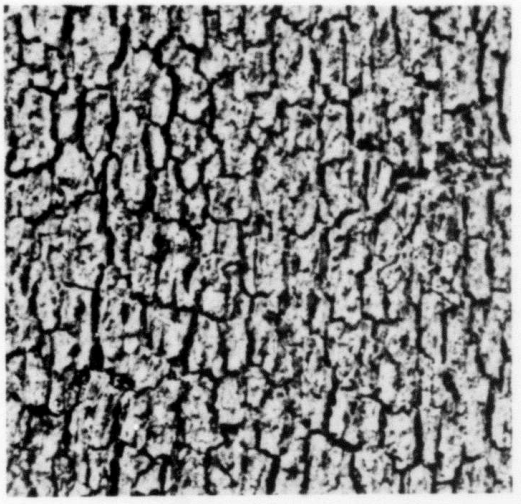


(b) Single linear model synthesis

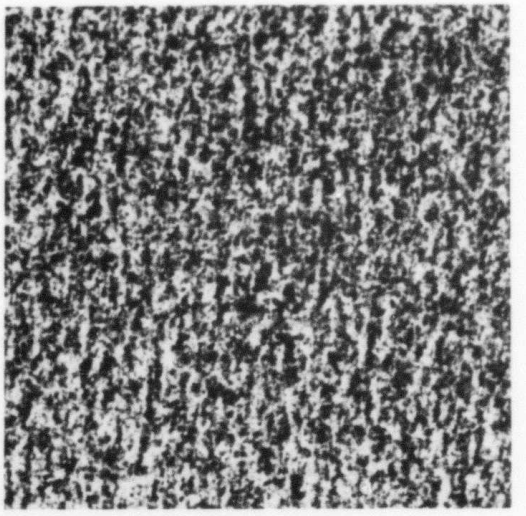


(c) Piecewise-linear model synthesis

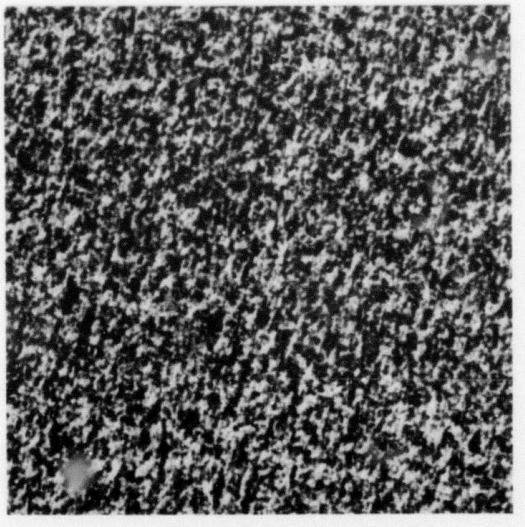
Fig. 2. Grass.



(a) Original parent texture

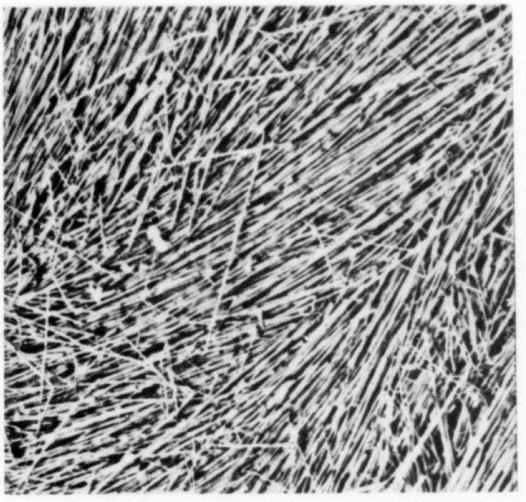


(b) Single linear model synthesis

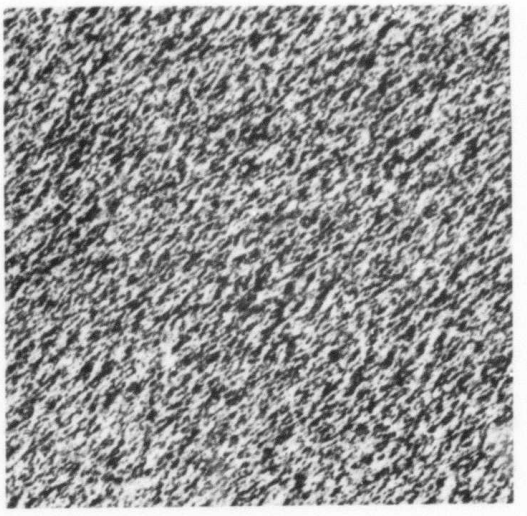


(c) Piecewise-linear model synthesis

Fig. 3. Bark.



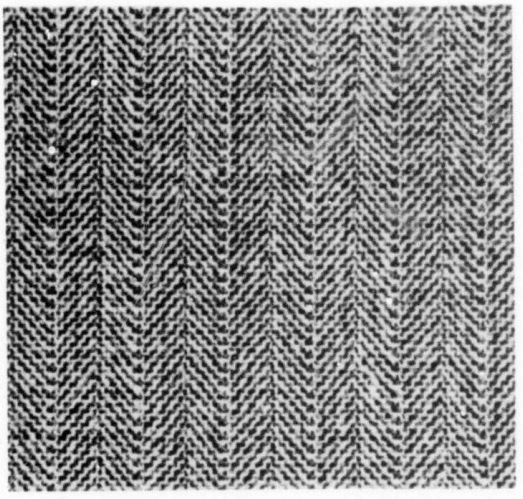
(a) Original parent texture



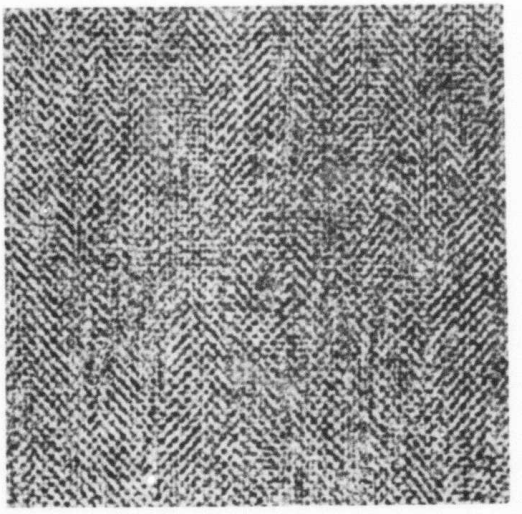
(b) Single linear model synthesis

(c) Piecewise-linear model synthesis

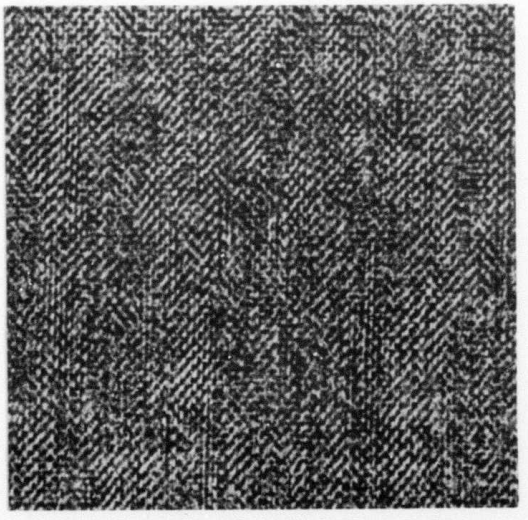
Fig. 4. Straw.



(a) Original parent texture

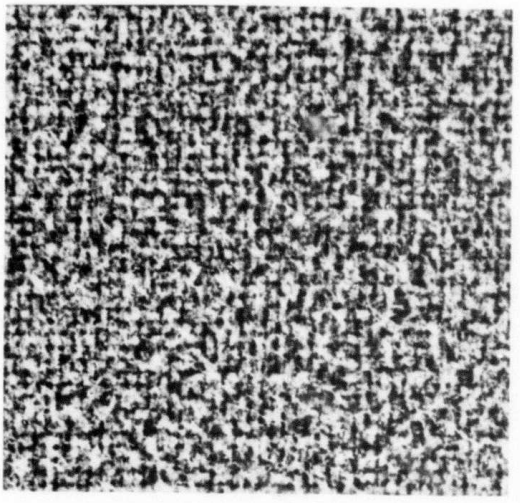


(b) Single linear model synthesis

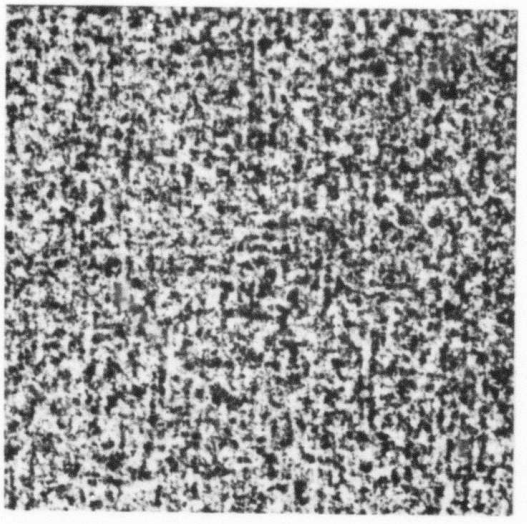


(c) Piecewise-linear model synthesis

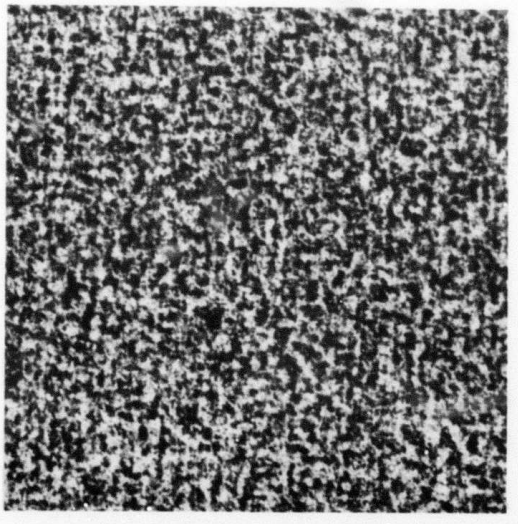
Fig. 5. Cloth.



(a) Original parent texture

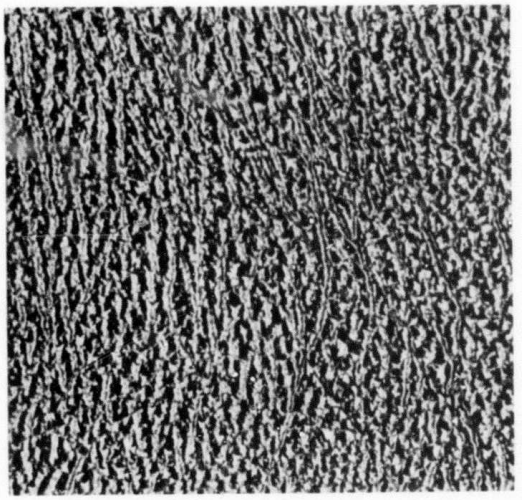


(b) Single linear model synthesis

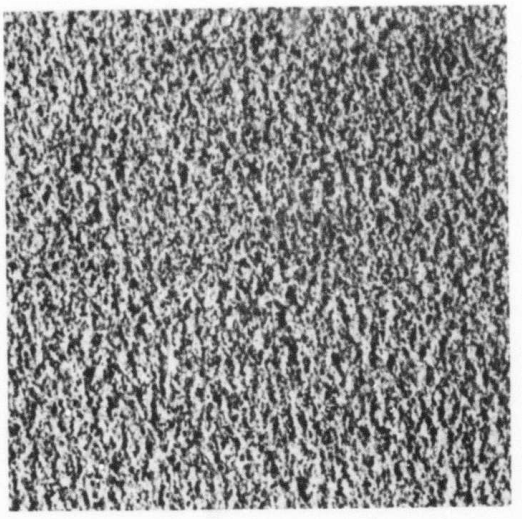


(c) Piecewise-linear model synthesis

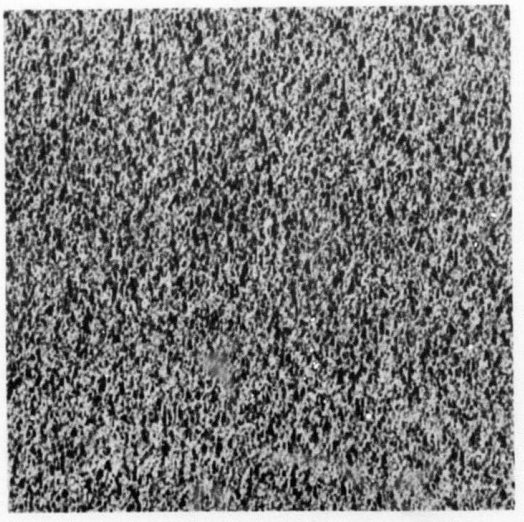
Fig. 6. Wool.



(a) Original parent texture

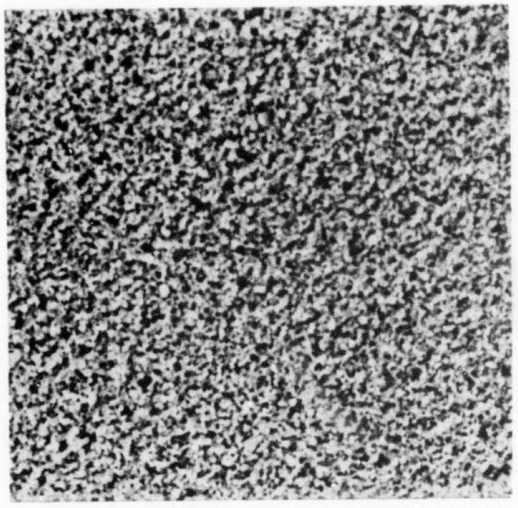


(b) Single linear model synthesis

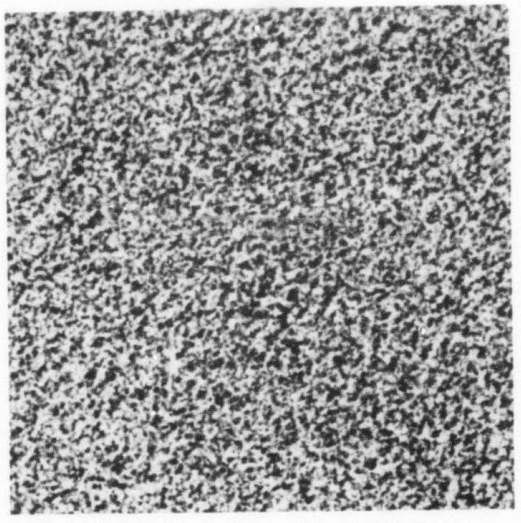


(c) Piecewise-linear model synthesis

Fig. 7. Leather.



(a) Original parent texture



(b) Single linear model synthesis

(c) Piecewise-linear model synthesis

Fig. 8. Sand.

what the example does not show is the potential instability of the β_i coefficient estimation for each of the line segments. As the number of models used increases, the range of the V_{N+1} being estimated by the model decreases, and the sample size used to estimate the coefficients is reduced. It is conjectured that this causes instability in the β_i estimation leading to usually-poor synthesis results.

In any case, it appears that the quality of the synthesis results do not warrant the expenditure of additional effort required to derive eight instead of one linear models.

REFERENCES

[1] D.D. Garber, "Computational Models for Texture Analysis and Texture Synthesis," USCIPI Report 100, Ph.D. Thesis, Electrical Engineering Department, University of Southern California, Los Angeles, May 1981.

[2] D.D. Garber and A.A. Sawchuk, "Additional Texture Synthesis Models and Results," USCIPI Report 990, 1980.

[3] D.D. Garber, "Models for Texture Analysis and Synthesis," USCIPI Report 910, 1979.

[4] D.D. Garber and A.A. Sawchuk, "Computational Models for Continuous-Tone Texture Synthesis," to be submitted to IEEE PAMI.

2.0 DEVELOPMENT OF VLSI IMAGE UNDERSTANDING SYSTEMS

S.D. Fouse, A.D. Cumming, V.S. Wong, G.R. Nudd

Hughes Research Labs

Malibu, CA.

INTRODUCTION

It is generally understood that image processing systems have very large computational throughput requiirements. (Typically greater than 25 million operations per frame.) If the system is to operate in real time (30 frames per second (FPS) then the required throughput is of the order of 1 billion operations per second. Obviously a special purpose processor is required to achieve real time performance, and because of the extreme computational requirements it is clear that Image Understanding systems can benefit greatly from the VLSI technologies. To be able to utilize VLSI, however, one must be able to overcome the anticipated high costs of design and test. One way to reduce the cost is to develop a processor which is very modular and can be described in a heirarchical manner, which is how the modern computer design tools will handle the complexity of a VLSI circuit. Another way to reduce the costs is to use regular structures on the chip, such as memory. This willl reduce design costs as well as the cost of testing.

Finally, probably the most significant way to handle the cost problem is to make a processor which has application to a wide range of systems. This will allow the cost of the chip to be amortized over a large user base.

For the last year we have been developing an LSI prototype of a VLSI processor which performs arithmetic operations over a sliding 5x5 window of an image. The RADIUS (Residue Arithmetic based Digital Image Understanding System) processor was described in the previous USC semiannual report [1]. This processor has several features that makes it very well suited to a VLSI implemetation, including modularity, extensive use of memory, and application to a large number of image understanding systems currently being developed.

As indicated by the acronym, the processor utilizes the technique of residue arithmetic [2] to perform the computations. The processor converts the incoming binary image data into a residue representation by calculating the MOD or remainder function over multiple, relatively prime bases. The data is then processed, in parallel independent channels, one for each base, with identical operations being performed for each base. In each channel the data is processed using modular arithmetic in the respective base. Finally the data from each base channel are combined to form a binary result.

Our prototype processor uses commercially available read-only-memories (ROMs) for the conversions from binary to

residue and residue to binary. For the computation portion of the processor we have developed an LSI nMOS circuit which can perform 5x1 local area computations for a single 5 bit base (<32). The processor, which uses 20 of these custom circuits (4 bases, 5 lines per base) is currently programmed to process data in the bases 31, 29, 23, and 19 and can accept 8 bit binary data at a 10 MHz rate.

This paper describes the critical aspects of the development of RADIUS and the progress that has been made. In addition we describe the status of the essential related projects we have been working on. These include:

- RADIUS-UNIBUS interface
- Applications of RADIUS
- Design Automation
- A Local Area Logic Processor.

PROGRESS ON RADIUS DEVELOPMENT

The RADIUS development work can be decomposed into three areas:

- Development of residue custom LSI circuit
- Fabrication of processor board
- System integration

Each of these are critical in that the system will not operate wihout the successful completion of the work in all three areas.

During the last six months we have made considerable contributions in all three areas.

The development of the residue custom circuit has significantly progressed in the past six months. In June,1981 the first parts were packaged and tested. Software was developed for testing all of the major components of the circuit including the input shift registers, RAM's, base latches, and modular adders. A problem was detected with the adders and soon diagnosed to be a design error. At this point we started a redesign of the circuit to correct the error as well as continuing to thoroughly test out the circuit for functional correctness and operating speed. The chip was run at 2.5 MHz with very clean waveforms. The limit on the speed was due to the bandwidth of the clock generator and clock drivers. In addition it was determined that there was only that single fatal error but several minor problems were also corrected in the redesign, which was completed in September, 1981. The new masks were produced and the processing of the new lot is due to be completed by the middle of October.

The actual processor consists of a 12x14 inch wire-wrap board. The majority of the wire-wrap was complete prior to the arrival of the first lot of chips. The board was tested without the chips and it was verified that all of the data paths were correct and the encoder, adder, and decoder ROMs were correct. When the chips did arrive, they were tested in the processor

itself. This provided the benefit of developing the processor programming software simultaneously with the test software. The current status of the processor board is that it is 95% complete with only fine tuning of the clocks remaining to be done. This will be done when the second lot of chips arrive.

The last area of work instrumental in the development of RADIUS is the integration of the custom chip, processor board, and external control. This work has proceeded in conjunction with the other two areas of effort, since all of the testing was accomplished through the use of the external control. The external control consists of a microcomputer with a 24 bit parallel I/O card. The programs that were written for test and programming have all been written in assembly code and execute on the microcomputer. This work is essentially complete except for insuring that the system is compatible with the new chips, and there is no reason to expect that the chips won't be compatible.

RADIUS-UNIBUS INTERFACE

The RADIUS processor is designed to accept data at 100 nanosecond intervals, which is fast enough for real-time stand alone operation. This means, however, that when the processor is used as a peripheral device attached to a general purpose computer, the data transfer will be limited by the memory cycle of the general purpose computer and not by the processor speed. The Hughes Image Understanding Installation is based on a PDP 11/34, and the fastest access to this is provided by a so-called

Direct Memory Access module- type DR11B. This is able to communicate with the computer memory along the UNIBUS lines, without intervention by the CPU. About 500,000 words per second can be transferred in this way, which translates to a data rate of 1 megabyte/sec at the RADIUS processor.

The 11/34 uses a page addressed memory structure, each page being 32k words in length. In order to store a complete image it is necessary to cross page boundaries by a technique known as dynamic region allocation. Software has been written to do this, which stores a 256x256x8 bit image buffer ready for processing. Our display unit, a COMTAL, can store two 512x512x8 bit images, each of which is split into four quadrants giving a further eight image buffers. It is anticipated that nine buffers in all should be adequate for the development of most of our Image Understanding algorithms.

The dynamic region allocation and direct memory access techniques will provide a string of 8 bit values, originally generated by a raster scan of the image. We will include in the processor interface the means for generating the two dimensional kernal. This kernel generation function is most easily envisioned as a series of shift registers. For a five line kernel four shift registers are required, each one containing as many elements as there are pixels on a line. However, since the system is being designed with variable line lengths a variable length shift register would be hard to implement. For this and

other reasons (including component availability, price, performance, reliability, etc) Hughes has developed a line delay system using random access memory. A system of address counters and crossbar switches is used to both load and retrieve the data on a line by line basis, and also to access the 5x5 kernel itself.

Another design issue in the computer to RADIUS processor interface is that of gaining access to the lookup tables carried in each of the 20 custom residue chips. This could be either done with a seperate interface module to the 11/34 or by using some of the existing control lines on the DR11B DMA module. We will take the latter course of action to avoid unnecessary clutter on the 11/34 backplane. This will, however, necessitate some extra switching on the interface module so that data from the DR11B can be routed to either the four line delay or to the lookup tables. For reasons described below, we intend to pass information to and from the RADIUS processor through lookup tables, composed of 256x8 bit RAMS. This will make it possible to apply nonlinear operations on a pixel by pixel basis, and will also control the dynamic range of signals fed to the RADIUS processor.

APPLICATIONS OF RADIUS

The primary motivation for developing this processor was to do 5x5 convolutions for such applications as edge enhancement, statistical differencing, low/high pass filtering, statistical

moment calculations, integer coefficient transforms, and texture analysis. However, the processor is capable of performing a much wider range of computations. The reason for this flexibility is due to the fact that we used a lookup table to perform a unary operation and the table is completely programmable. The general form of the computation that can be performed by the processor is:

$$y = \sum_i f_i(x)$$

where y is the output, xi are the 25 elements in the 5x5 kernel, and fi are any allowable residue arithmetic computations.

The exact nature of what constitutes an "allowable residue arithmetic computation" is an interesting one and we are performing ongoing studies of this. Basically, the integer operations of addition, subtraction, and multiplication are allowed, but division is not, since the result of a division is not, in general, an integer. In the residue arithmetic as implemented on the RADIUS machine, any number is uniquely represented by an array of four residues in bases 19, 23, 29, and 31. If any addition, subtraction, or multiplication operation is performed on all four residues (modulo the base in question), that is equivalent to performing the same operation on the input number. The largest number that can be represented, M, is given by the product of the bases, in this case 392863. Overflow cannot occur in any one base since the answer is always taken modulo the base, but it is possible for the output number to

exceed M. In this case, the computed result appears modulo M, giving a 'wrap around' effect similar to that in binary arithmetic. However, there is no convenient way to determine that this overflow has occurred, necessitating considerable care in developing an algorithm to generate fi(xi). (It is imperative that the final result of the algorothm does not exceed M). An advantage of residue arithmetic, however, is that intermediate values in a calculation can be arbitrarily high, and the algoithm can be arbitrarily complex. This arbitrary complexity is a very powerful feature of residue arithmetic, and it arises because the results are stored in a lookup table. However much computer time is needed to generate the tables themselves, the computations are still performed on image data at the full 10 MHz rate.

Since the RADIUS processor can evaluate any arbitrary integer coefficient polynomial and many useful operations can be approximated by polynomial functions, we have decided to investigate this approach. The polynomial may be of arbitrarily high order with arbitrarily large coefficients, although the coefficients must be integer. In particular, no coefficient can be less than one so that in a high order term, a large input number raised to a high power may exceed the maximum M. This is not a problem if other terms in the polynomial are sufficiently negative to reduce the overall result to less than M, and we are working on polynomial curve fitting techniques to achieve this. Figure 1 shows an approximation to the function $y=a\sqrt{x}$, useful as part of the Sobel operator. As may be seen an accurate fit is

obtained with a cubic polynomial everywhere except the origin, and it is possible to improve the fit at the origin if necessary by adjusting the polynomial coefficients. There is a tradeoff between accuracy of the fit and the maximum allowable input value, and this is rapidly improving as we improve our curve fitting procedure. The preliminary results shown in Fig. 1 was produced simply by performing a least squares fit and then rounding the coefficients to the nearest integer value, but this is far from being the optimal technique. It is possible to fit a polynomial curve to practically any desired function, so that such operations as contrast enhancement, thresholding, etc., become possible. Furthermore, it is possible to perform a different function for each element of the kernel, so that the RADIUS processor is expected to open a new vista in convolution-type processing algorithms.

DESIGN AUTOMATION

This work is not directly part of the I.U. effort but will be of great benefit in terms of cost and speed of our designs. We have set ourselves the goal of investigating and implementing a design automation system capable of designing large VLSI chips in an efficient and reliable manner. If the predicted downscaling of devices and increase of chip density materializes in the near future, we will need much better design tools to organize and design VLSI chip systems, since we see that design productivity and system complexity will be the bottlenecks in

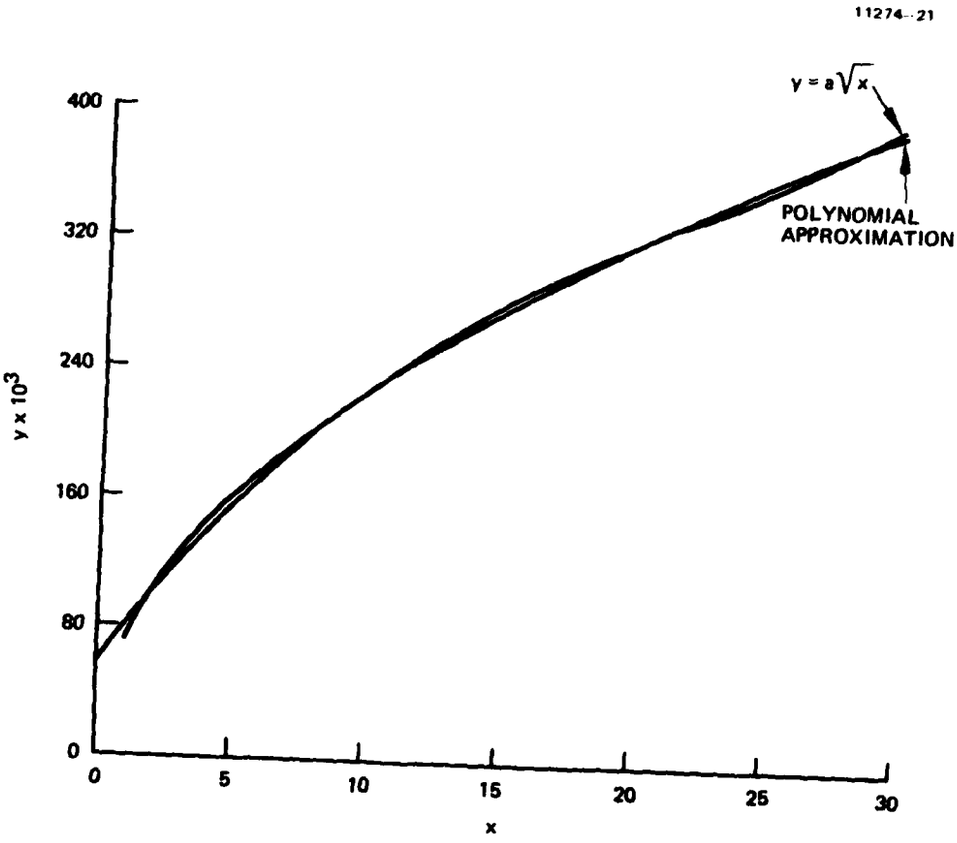


Figure 1. Scaling Factor Chosen to Utilize Dynamic Range of Residue Processor

implementing VLSI systems.

To help us develop these tools for VLSI design, we have at our disposal several computer systems and devices. These include a VAX 11/780 where we are developing most of our software, an AVERA IC designer which is a complete IC design system in itself, a PDP 11/34 which is used as a controller, and several plotters and graphics terminals. We also have access to an Amdahl 470 processor and a PDP 10 computer for running simulations and for program development.

The AVERA unit is a self contained IC design system including layout graphics and symbolic representation of designs. It includes dual floppy disks where designs and system programs are kept, a 17 in. black and white CRT, a keyboard, and digitizing tablet. The capabilities include symbolic recognition of commands and up to 64 levels of design representations. The design output can either be in CALMA GDS II or CIF format.

We are also investigating the design automation approaches being pursued at the Universities. One example is a STICKS-type design package called CABBAGE developed at UC Berkeley. It enables a designer to use symbolic representations to formulate his designs, and as a final step, compacts the design to minimum geometry while observing the programmed design rules. Another approach being taken at Caltech and MIT is to specify the designs in a structured and algorithmic way. Standard VLSI components such as PLAs and ALUs can be designed in this way with alterable

parameters controlling the size and configuration of the component. Other groups have looked at the testing and simulation of complex VLSI system designs. In order to design error free systems it is necessary to be able to simulate complex chip designs. Research on this is being done at MIT, where they have developed a program for logic simulation called MOSSIM.

A LOCAL AREA LOGIC PROCESSOR

RADIUS has wide application to image understanding and can perform the vast majority of arithmetic operations required. There is, however, a need for additional high-speed processing at the pixel level for operations such as those requiring logical decisions. Examples of the need for this type of processing are operations such as edge thinning, edge tracing, and region formation. Recognizing the need for this type of high speed processing, we are working to define and develop a logic processor to complement RADIUS.

The first step in the development of the logic processor is to define an instruction set. This set of instructions should allow a wide range of functions to be performed on an image, in single and multiple passes. The processor will access a local neighborhood of each pixel and will produce an output based on a comparison of the neighborhood to a template or a set of conditions. This concept is illustrated for three image frames in Fig. 2. The development of instructions or constraints required for the logic processor are not difficult to develop and

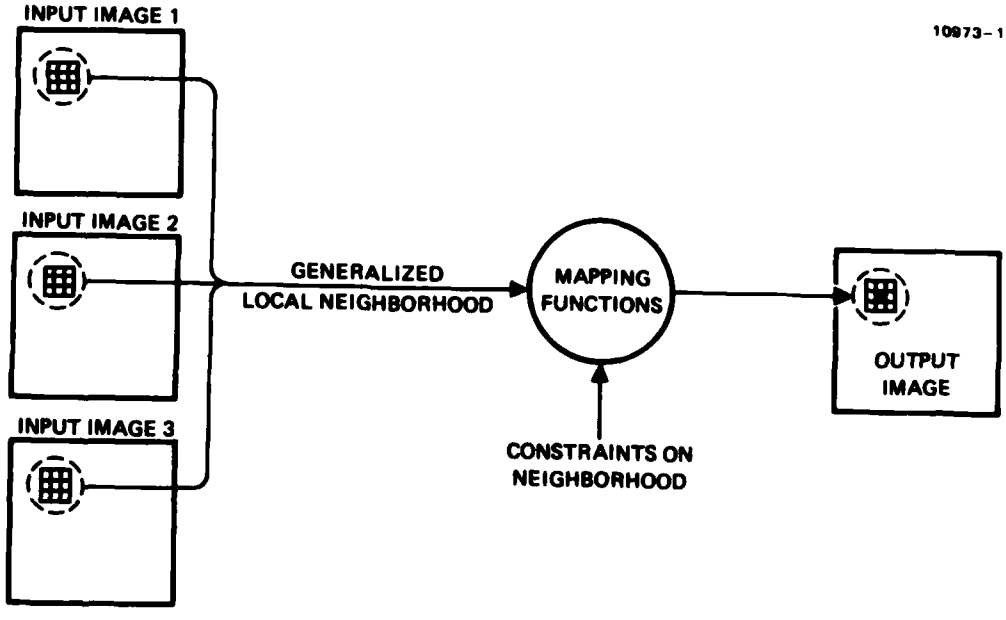


Figure 2. Inputs and Outputs for Logic Processor

have been generated for processors such as PICAP [3]. What we shall aim for is an efficient set of operations to allow high speed performance, matched to the RADIUS machine. The type of neighborhood that is accessed, the types of conditions that can be specified, and the types of mappings from the combination of local neighborhood and conditions to an output pixel determine the instruction set parameters and specify a minimum capability for the processor.

Once the processor instruction set is defined, we will determine an appropriate architecture. As with the RADIUS processors we will look towards architectures that will be suited to VLSI implementation. For the same reasons that motivated the design of RADIUS, we will probably utilize look-up tables to perform some of the operations, making good use of memory structures which are easily designed.

A high level block diagram of a processor which could probably perform the range of operations required of a logic processor is shown in Fig. 3. This processor is comprised of logic to form the local neighborhood, which can be controlled to perform comparisons on the data in the neighborhood, and two look-up tables to provide the mapping from the results of the comparisons and the center pixel to an output pixel. The look-up tables, in addition to making a chip easy to design, will also give the processor a great deal of flexibility. As a benchmark example, we have investigated the median function. Using this

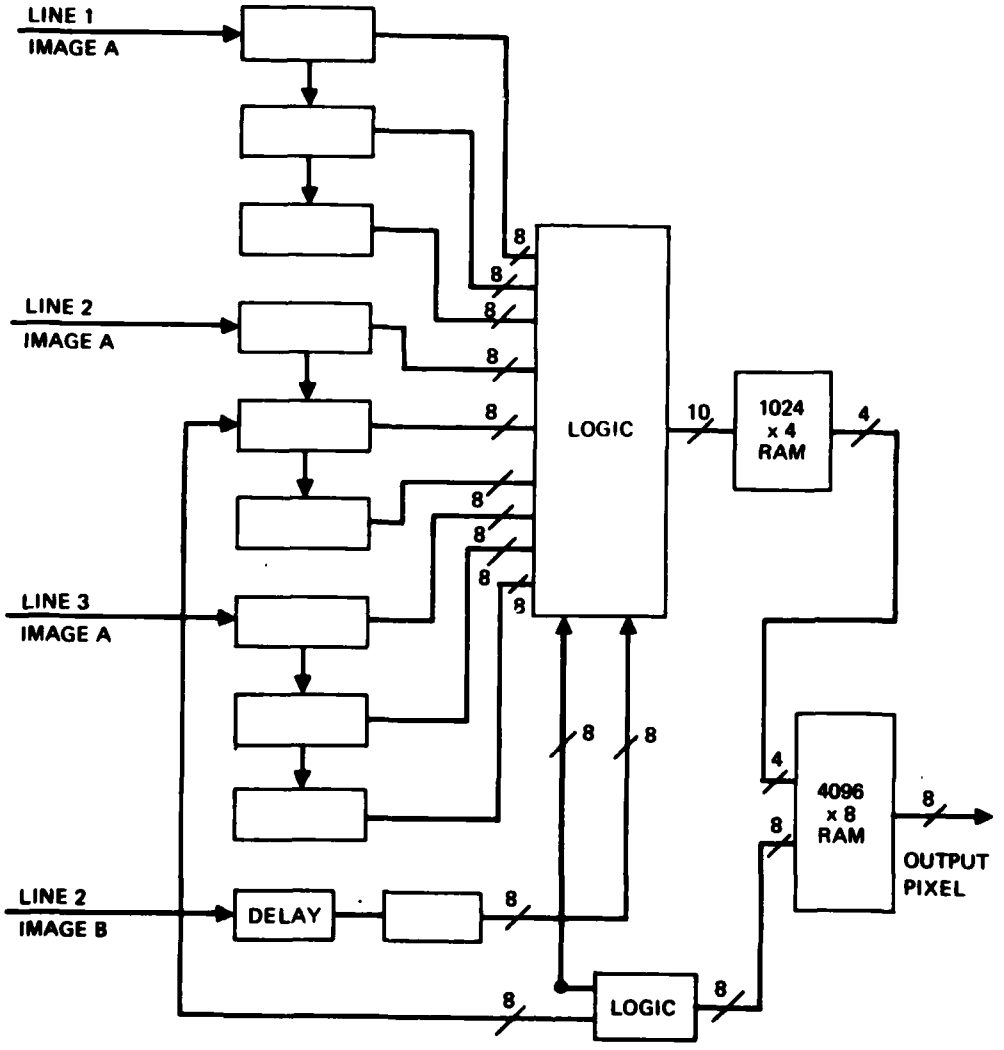


Figure 3. An Architecture for a Local Area Logic Processor

architecture, the processor would be capable of performing a median calculation of some local neighborhood with only 8 passes. This is just an indication of the power a table driven processor could provide.

SUMMARY

We have described ongoing work on the developmenT of RADIUS, a real-time image understanding system which is well suited to VLSI. The prototype system is due to be operational in November, 1981. In addition to the development work we have described, we are working on several related tasks, including developing an interface between a PDP 11/34 and the RADIUS processor, investigating further applications of residue arithmetic to image understanding, and developing an integrated design automation capability that will allow us to design and simulate LSI and VLSI circuits.

Our future work will include further development of the RADIUS processor and the development of the local area logic processor we described. When they are both complete we will have an integrated pixel level processor that can perform a wide range of functions in real-time and many more functions in near real-time. We will also develop an interface to the PDP 11 for the intergrated RADIUS-LOGIC processor, and this will allow all pixel operations to be performed at high speed, reducing greatly the CPU time needed for image understanding programs.

REFERENCES

[1] S.D. Fouse, G.R. Nudd, G.M. Thorne-Booth, P.A. Nygaard, and F.D. Gichard, "A Residue Based Image Processor For VLSI Implementation," USCIPI Report 1010, March, 1981, pp.73-98.

[2] N. Szabo and R. Tanaka, Residue Arithmetic and its Applications to Computer Technology, McGraw-Hill, New York, NY, 1967.

[3] B. Kruse, "System Architecture for Image Analysis," Chapter 7 of Structured Computer Vision, edited by S. Tanimoto and A. Klinger, Academic Press, 1980.

3. RECENT PUBLICATIONS AND PRESENTATIONS

[1] B. Bhanu, "Shape Matching and Image Segmentation Using Stochastic Labeling," Ph.D. Thesis, published as USCIPI Report 1030, Department of Electrical Engineering, University of Southern California, Los Angeles, California, August 1981.

[2] B. Bhanu and O.D. Faugeras, "Segmentation of Images Having Unimodal Distributions," accepted for publication, IEEE Trans. on Pattern Analysis and Machine Intelligence.

[3] B. Bhanu and O.D. Faugeras, "Recognition of Occluded Two-dimensional Objects," Proc. 2nd Scandinavian Conf. on Image Analysis, Helsinki, Finland, June 15-17, 1981, pp. 72-77.

[4] B. Bhanu, "Computation of Two-dimensional Complex Cepstrum," submitted to the IEEE International Conference on Acoustics, Speech and Signal Processing, Paris, France, May 3-5, 1982.

[5] D.D. Garber, "Computational Models for Texture Analysis and Texture Synthesis," Ph.D. Thesis, published as USCIPI Report 1000, Department of Electrical Engineering, University of Southern California, Los Angeles, Ca., May

1981.

[6] D.D. Garber and A.A. Sawchuk, "Texture Simulation Using A Best-Fit Model," Proc. of 1981 IEEE Computer Society Conference on Pattern Recognition and Image Processing, pp. 603-608, Dallas, Aug. 3-5, 1981.

[7] D.D. Garber and A.A. Sawchuk, "Texture Simulation Using a Best-Fit Model," submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.

[8] D.D. Garber and A.A. Sawchuk, "Simulation of Natural Binary Textures Using Nth-order Joint Density Functions and Linear Models," submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.

[9] D.D. Garber and A.A. Sawchuk, "Computational Models for Continuous-Tone Texture Synthesis," submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.

[10] O.D. Faugeras and B. Bhanu, "Reconnaissance de Formes Planes Par Une Methode Hierarchique d'etiquetage probabiliste," Proc. AFCET, 3eme Congres, Reconnaissance des Formes et Intelligence Artificielle, Sept. 1981, Nancy, France.

[11] T.C. Henderson and B. Bhanu, "Extraction of Planar Faces from Range Data," submitted to the International Workshop on Industrial Applications of Machine Vision, Raleigh,

N.C., May 3-5, 1982.

[12] D. Kuan, P. Chavel, A.A. Sawchuk, and T.C. Strand, "Discrete Speckle Image Modeling and Restoration," 1981 Annual Meeting Optical Society of America, Orlando, Florida, October 1981, Journal of the Optical Society of America, Vol. 71, (December 1981).

[13] D. Kuan, P. Chavel, A.A. Sawchuk, and T.C. Strand, "Nonstationary Two-Dimensional Recursive Image Restoration," 1981 Annual Meeting Optical Society of America, Orlando, Florida, October 1981, Journal of the Optical Society of America, Vol. 71, (December 1981).

[14] R. Nevatia, "Image Understanding Research at USC," Seminar at the USC Information Sciences Institute, July 1981.

[15] R. Nevatia and A.A. Sawchuk, "Progress in USC Image Understanding Program," Proc. of ARPA Image Understanding Workshop, Washington, D.C., April 1981, pp. 241-242.

[16] A.A. Sawchuk, "Modeling of Signal-Dependent Noise," Second ASSP Workshop on Two-Dimensional Signal Processing, New Paltz, New York, October 1981.

[17] B.H. Soffer, J.D. Margerum, A.M. Lackner, D. Boswell, A.R. Tanguay, Jr., T.C. Strand, A.A. Sawchuk, and P. Chavel, "Variable Grating Mode Liquid Crystal Device for Optical Processing and Computing," Molecular Crystals and

Liquid Crystals, Vol. 70, pp. 145-161, (1981).

[18] A.R. Tanguay, Jr., T.C. Strand, P. Chavel, A.A. Sawchuk, and B.H. Soffer, "Theoretical and Experimental Polarization Properties of the Variable Grating Model Liquid Crystal Structure," 1981 Annual Meeting Optical Society of America, Orlando, Florida, October 1981, Journal of the Optical Society of America, Vol. 71, (December 1981).

[19] F. Vilnrotter, "Structural Analysis of Natural Textures," Ph.D. Thesis, published as Report ISG 100 and USCIPI 1040, Department of Computer Science, University of Southern California, Los Angeles, California, September 1981.

[20] F. Vilnrotter, R. Nevatia and K. Price, "Structural Analysis of Natural Textures," Proceedings of ARPA Image Understanding Workshop, Washington, D.C., April 1981, pp. 61-68.